ELEKTOR ELECTRONICS

APRIL 2003
£3.45

THE ELECTRONICS & COMPUTER MAGAZINE

www.elektor-electronics.co.uk

8-Channel Disco Light Controller

Switched-Outlet Power Bar

Freeware C Compilers

USB-RS232 Interface

Week/ Month Timer

Subsonic Microphone

VALVE FINAL AMP

35 watts from two EL34s

9 770268 451104 04

Valve Final Amp

35 watts from a no-frills design

Design by Bob Stuurman

This valve power amplifier is a push-pull design using two EL34s (or their 6CA7 US equivalents). It has been kept fairly simple to avoid problems with DIY construction. The output power is well above 35 watts, with low distortion and a wide frequency range. This amplifier provides excellent sound reproduction when used with a pair of reasonably efficient, good-quality loudspeakers — and it shows that even a simple design with quite conventional specifications can sometimes make you tremble with excitement when listening to certain musical passages.

This final amplifier is based on a Philips design dating from the late 1950s, with a few modifications suggested by Claus Byrith. These modifications consist of a separate supply for the negative grid voltage for the EL34s, an AC balance adjustment for the output stage, an EF86 pentode wired as a triode in the preamplifier stage and a reduction in the amount of overall negative feedback (20 dB). Two documents on this subject have been published on the Internet. They describe the design in detail, and they are certainly worth reading if you are interested in this topic (see 'References').

Since the actual circuit is well documented, we limit ourselves to a brief description in this article. However, we do have a bit more to say about some of the less well-known aspects of the design, because they provide good insight into the problems associated with push-pull valve final amplifiers and the available solutions.

In the first part of the article, we address the theoretical aspects of the design, and in the second part we turn our attention to its construction. Since this is a DIY project, rather than a kit, certain parts of the construction are described in fairly extensive detail.

Schematic diagram

Figure 1 shows the complete schematic diagram of a single channel of the Valve Final Amplifier. There are three supply voltages: a positive high voltage of +440 V, a negative grid voltage of –55 V and a filament voltage of 6.3 V. Separate filament circuits are used for the preamplifier/phase splitter (Fil1 & Fil2) and the output valves (Fil3 & Fil4). The filaments are symmetrically connected to circuit ground via R28 and R29.

The output valves are operated in the 'ultralinear' mode by connecting their screen grids to taps on the anode windings of the output transformer via 1kΩ resistors. Due to the internal negative feedback via the screen grids, the pentodes exhibit characteristics lying between those of a triode and those of a normal pentode. Their internal impedance is reduced to practically the same level as that of a triode, and distortion is reduced to the triode level. However, the output power also drops to around 65 percent of that provided by a pure pentode output stage.

Instead of obtaining the negative grid voltage for the output valves from a voltage drop across the cathode resistors, we use a separate grid voltage supply. This prevents the operating point of the valves from shifting during operation. The magnitude of the negative grid voltage for the output valves can be adjusted using P2 ('DC current'), while the DC balance can be adjusted using P3.

The output stage operates in Class A for small signals, but it shifts increasingly towards Class B as the signal level increases. The current consumption also increases with larger signals. The operating point can be set within certain limits by adjusting the magnitude of the negative grid voltage. Since a separate supply is used for the negative grid voltage, the full anode supply voltage is present across the output valves.

The cathodes are connected to signal ground via 10 Ω resistors (R24 and R25). The voltages across these resistors are proportional to the currents through the valves (10 mV/mA).

Three test points are provided for aligning the circuit. TP0 is circuit ground, while TPV3 and TPV4 are the alignment test points for valves V3 and V4, respectively.

The EL34s provide maximum output power when the voltage on the control grid is

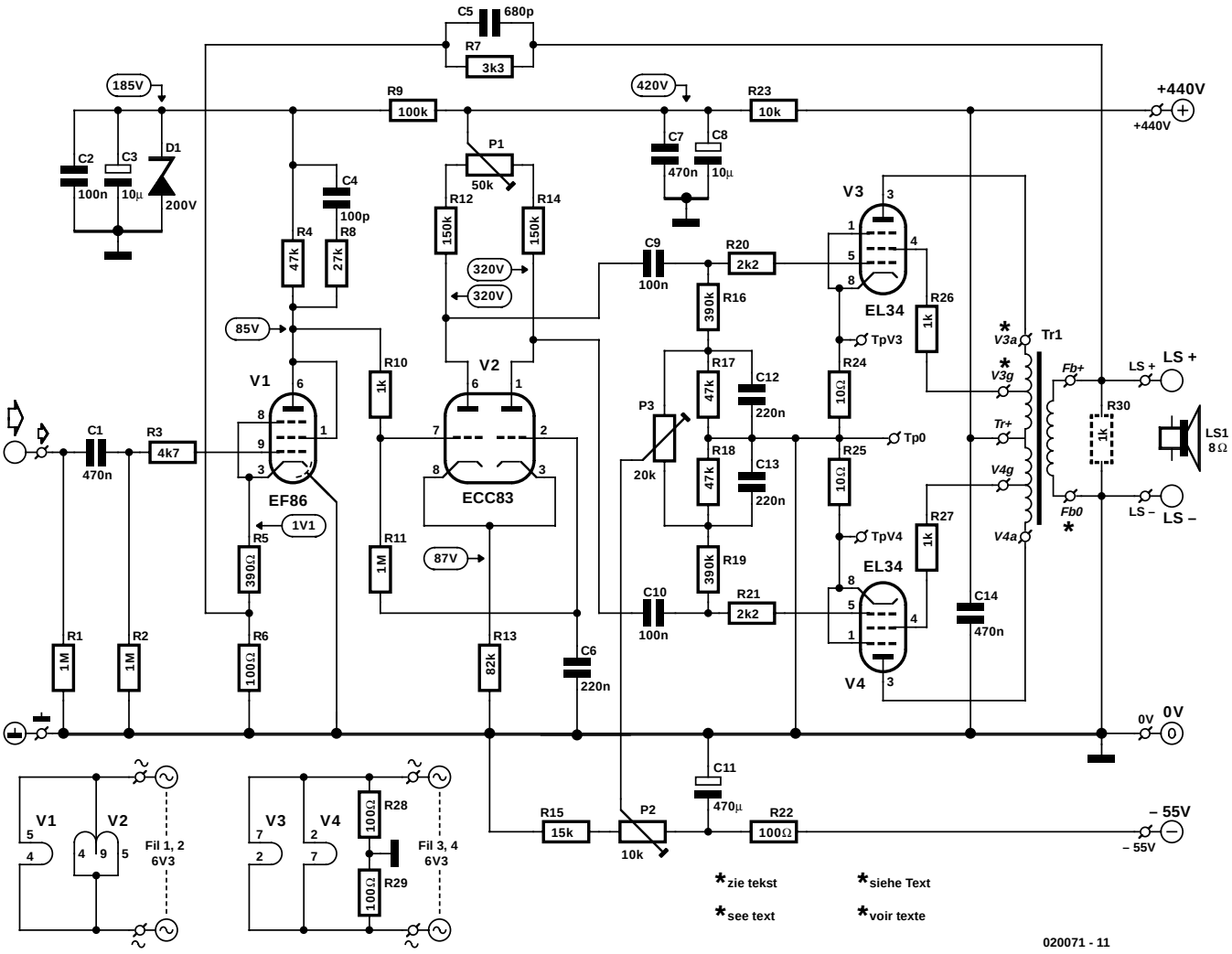


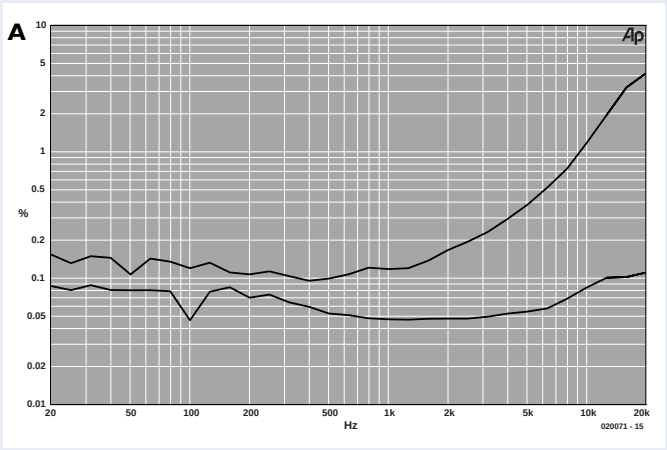
Figure 1. Schematic diagram of the Valve Final Amp.

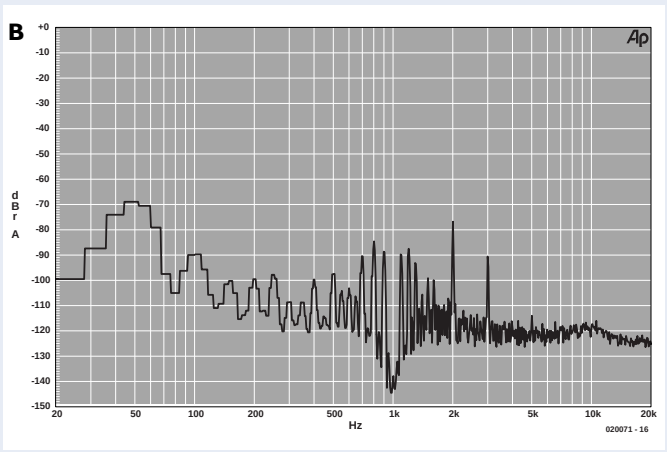
Amplifier Specifications

Input impedance:	1 MΩ
Input sensitivity:	600 mV
Nominal loudspeaker impedance:	8 Ω (4 Ω optional)
Maximum output power:	39 W into 8 Ω
Bandwidth at 1 W:	5 Hz – >40 kHz
THD+ noise (1 W/8 Ω, 1 kHz):	0.06% (B = 80 kHz)
Signal to noise ratio:	62 dB (B = 22 kHz)
	88 dB (A-weighted)

Performance

A few measured results are shown here. **Plot A** shows harmonic distortion versus frequency. The lower curve was measured at an output power level of 1 W, and the upper curve at 27 W. The 1 W curve in particular is very nice, and this is a typical power level for listening to music. **Plot B**, which is rather more irregular, shows an FFT analysis of a 1 kHz signal at an output power of 1 W. The 1 kHz sine wave has been suppressed by the measuring equipment, and the remaining peaks represent the distortion residuals of the amplifier. You shouldn't be overly alarmed by this plot, since the very wide dynamic range of the analyser (150 dB) gives an exaggerated impression of the actual situation. The most important components are the distortion peaks at 2 kHz and 3 kHz, which lie at –77 dB and –90 dB, respectively. For a relatively simple design using valves and transformer output, this is a very good result. The bulge at 50 Hz results from residual hum in the supply voltage and has nothing to do with the distortion spectrum.





approximately 26 V. This drive level can be easily provided by the phase splitter. The phase splitter is a type having the cathodes connected together and the grid of the second triode (V2b) grounded for AC signals by C6. Since V3a is driven by the grid and V2b is driven by the cathode, there is a small imbalance in the magnitudes of the AC voltages on the anodes. The voltages can be adjusted to be exactly the same using P1 ('AC balance').

The phase splitter exhibits a gain of approximately 26 times, so a signal level of 1 V on the grid of V2a is needed to fully drive the output stage. The high resistance of the cathode resistor (R13) yields low distortion and a high cathode voltage (around 87 V), thus allowing the grid of V2a to be driven directly from the anode of the EF86 preamplifier valve without using a coupling capacitor.

The preamplifier is wired as a triode by connecting the screen grid to the anode, since the high gain that can be obtained with a pentode is not needed. This reduces the noise factor to that of a triode, while retaining the good internal screening and freedom from microphonics characteristic of this valve.

A signal level of 60 mV on the grid of the EF86 is needed to fully drive the output stage. Due to the 20 dB of negative feedback provided by R7 and R6, the input level needed to fully drive the output stage is 600 mV. At this level, the output power is 39 W. The amplifier starts to clip at an input level of 0.7 V, which corresponds to an output power of around 46 W.

The resonant frequency of the output transformer due to its leakage inductance is approximately 80 kHz. At this frequency, the open-loop gain must be small enough to ensure that the amplifier remains stable. The necessary gain roll-off is provided by C4 and R8, with a bit of help from C5. The values of these components were determined experimentally using square-wave signals.

When the amplifier is switched on, the high voltage and negative grid voltage are present almost immediately. However, the filaments must warm up before any current can flow through the valves. Diode D1 is thus included to prevent an excessively high voltage from appearing on the anode and screen grid of the EF86. The circuit reaches its normal operating state after a few tens of seconds, with a voltage of approximately 185 V across D1.

RF-suppression ('stopper') resistors are used for the control grids of all the valves. They were present in the original design, so we have kept them here as well.

In the original design, the screen coupling capacitors for the output valves (C9 & C10) had a value of 470 nF. The current through the output valves proved to have rather large fluctuations at a very low frequency (0.2–0.5 Hz), which were also present at the loudspeaker output. This was probably due to small variations in the negative grid voltage. Since these fluctuations have a small amplitude and the output transformer has a large self-inductance, they are not blocked by the output transformer, and they find their way to the amplifier input via the negative-feedback network. This phenomenon was reduced to an accept-

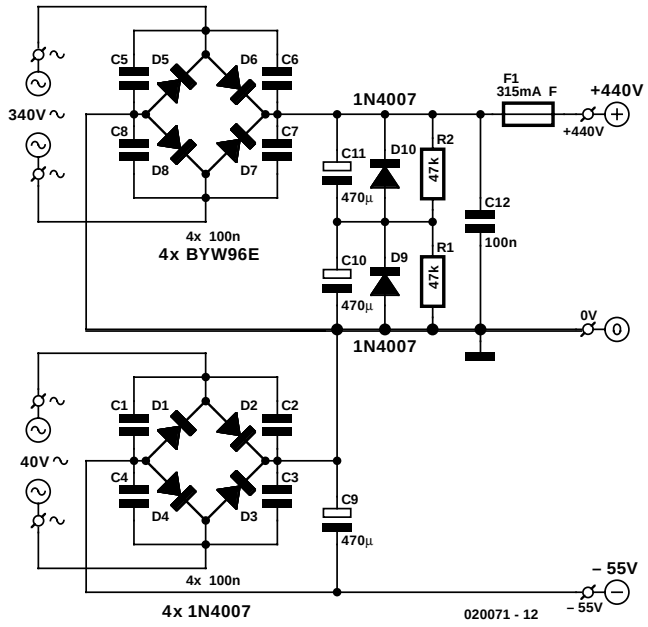
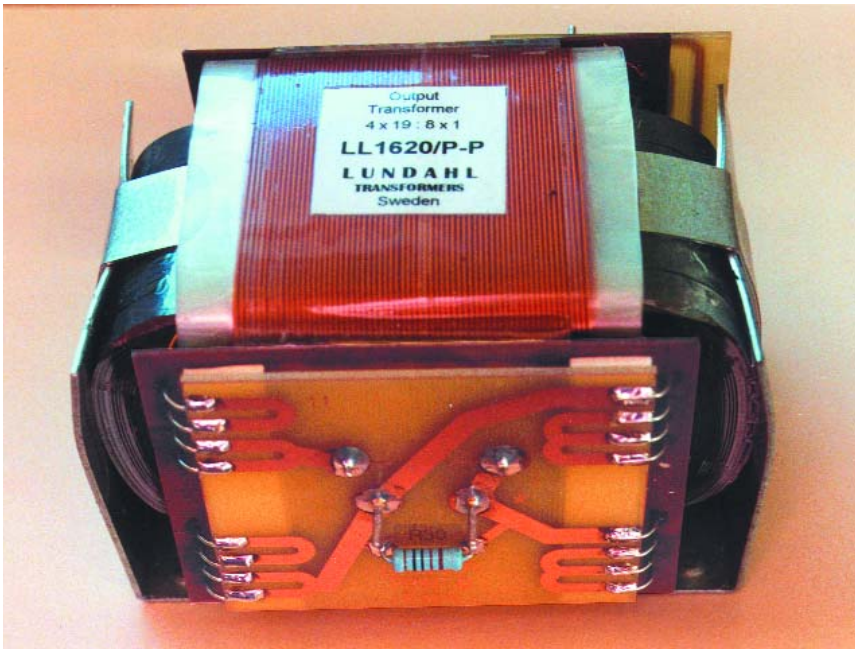


Figure 2. Schematic diagram of the power supply.



able level by decreasing the value of C9 and C10 to 100 nF. This does not have any audible effect on the reproduction of low frequencies.

Power supply

The good characteristics of the Valve Final Amp are in part due to its robust power supply. The Amplimo type 7N607 toroidal transformer, which weighs around 3.5 kg, can provide 340 V at a healthy 700 mA. After rectification and filtering, more than 400 mA at 440 V are available to the amplifier. The winding for the negative screen voltage provides 40 V at 100 mA, which yields an adequate voltage (55 V) after rectification and filtering. The total filament current of the valves is about 7 A. The 6.3-V winding is rated at 6.8 A, but since the load on the high voltage winding is fairly small and practically no power is drawn from the screen-voltage winding, this does not present a problem.

Figure 2 shows the schematic diagram of the power supply. The high voltage is rectified by four diodes wired in a bridge configuration. The diodes have a surge current rating of 60 A. Interference suppression ('anti-rattle') capacitors are connected across the diodes. Since it is practically impossible to buy high-voltage electrolytic filter capacitors with large capacitance, a pair of 470 µF/400 V electrolytic capacitors are connected in series to achieve an effective capacitance of 235 µF. Diodes D9 and D10 prevent the capacitors from being reverse-biased when the amplifier is switched off. Resistors R1 and R2 divide the voltage evenly across the capacitors and discharge them within several minutes after the amplifier is switched off. C12 provides RF decoupling. Protection is provided by a 315-mA, fast-acting (F) fuse, and it can be a lifesaver for the output valves if the negative grid voltage becomes too small (less negative).

Output transformer

The most important, most critical and invariably most difficult to obtain component of a push-pull valve amplifier is the output transformer. The original Philips design used an output transformer having ten primary windings connected in series, with eight secondary windings interleaved between the primary windings. The secondary windings could be connected in a series/parallel arrangement to obtain the desired input and output impedances. This must have been a real whopper of a transformer, and we estimate that it surely must have weighed more than 5 kgs.

You may be wondering why it was necessary to use a transformer wound in such a complicated manner. The reason is that the ability of a transformer to pass a sine-wave signal decreases as the frequency of the signal increases. Even with very good transformers, the drop-off at 25 kHz is already around 0.5 dB.

Figure 3 shows the equivalent circuit of a transformer driven by an electronic valve. Part (a) shows the situation at very low frequencies. Here the self-inductance of the primary must be high in order to limit the current and allow sufficient magnetic flux to be generated without going into saturation. Part (b) shows the situation at mid-range frequencies, where is a high impedance. Part (c) shows the situation at high frequencies, where the signal is attenuated by the leakage inductance (L_s) and the interwinding capacitance (C_w). The leakage inductance arises from the 'leakage' of magnetic flux as a result of incomplete coupling between the windings.

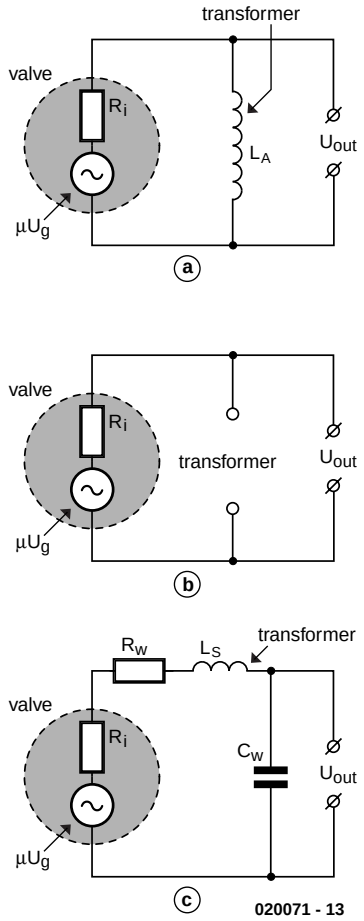


Figure 3. Output transformer equivalent circuit at various frequencies.

It takes time for a signal to pass through a transformer, since the low-pass filter formed by the leakage inductance and load impedance creates a time delay. The resulting phase difference between the input and output signals increases with increasing frequency. The output signal thus lags further and further behind the input signal as the frequency increases At 20 kHz, the phase difference can already be 14 degrees. Needless to say, this can have serious consequences for the reproduction of rectangular signals. Fortunately, there is a technique that can be used to deal with the problems of attenuation of high-frequency signals and increasing phase difference at higher frequencies. This technique is negative feedback.

Returning to the output transformer (see Figure 3), we see that L_s and the C_w also form a resonant circuit, so a rapid increase in the phase angle occurs when the signal frequency passes through the resonant frequency of this circuit. This can make the amplifier unstable. Consequently, the open-loop gain of an amplifier with negative feedback must be attenuated such that the gain–feedback product ($A \times \beta$) is less than 1 at this frequency. If the amplifier is to have a wide bandwidth, it is thus essential for the output transformer to have a sufficiently high resonant frequency. This requires the leakage inductance and winding capacitance to be small, which can only be achieved using complicated winding methods — such as the method used in the previously mentioned Philips output transformer. Naturally, such a transformer cannot be inexpensive.

After some searching, we found a valve output transformer that appears to be eminently suitable for the modified Philips amplifier design. This is the type LL1620PP transformer from the Swedish company Lundahl. This transformer has a 'C' core made from a special type of iron, with two primary windings and four secondary windings on each leg. The two halves of the core are held tightly together on the transformer frame by a welded ribbon. The push-pull version of this transformer (versions for use in single-ended amplifiers are also available) has a small (25 µm) air gap, so a slight imbalance in the DC currents through the primary windings can be tolerated without causing a large reduction in the primary self-inductance. The four primary windings are connected symmetrically in series, yielding taps at the 50-percent points the of windings that can be connected to the screen grids of the pentode output valves for operation in the 'ultralinear' mode. The eight secondary windings can be connected in series and/or parallel in various manners in order provide an output impedance of 4 Ω or 8 Ω. At 13 mH, the leakage inductance of the LL1620PP is somewhat on the high side, but this is inevitable with such a large primary self-inductance (no less than 300 H). Since the open-loop gain and negative feedback have both been reduced in the modified version of the amplifier, it remains stable despite the relatively leakage inductance.

The most important specifications of the transformer are listed in the 'Basic LL1620PP Specifications' box. The transformer dimensions are shown in **Figure 4a**. Paxolin boards with leads numbered as shown in the figure are fitted on both sides of the windings. The winding diagram of the transformer is shown in **Figure 4b**. Each primary winding is sandwiched between two secondary windings.

To make it easier to use the transformer and reduce the chance of wiring errors, the author has designed three small printed circuit boards for making connections to the transformer. They are not available from Readers Services, but if you want to make them yourself, you can download the layouts from the *Elektor Electronics* website (Free Downloads, reference number 020071-1, month of publication). However, it is certainly not difficult to wire the transformer into the circuit by hand. The necessary connections are shown next to each of the circuit board layouts.

For each of these circuit boards, the transformer is located on the 'component side' of the board. The numbers on the boards (1, 8 and 11) correspond to the lead numbers of the

Basic LL1620PP Specifications

Primary/secondary turns ratio:	4 x 19.2 / 8 x 1
Primary winding DC resistance: *	308 Ω (4 x 77 Ω)
Secondary winding DC resistance: (average per winding)	0.4 Ω
Primary winding self-inductance:	300 H
Primary winding leakage inductance: *	13 mH
Primary impedance in this design:	6 kΩ
Secondary impedance in this design:	4 Ω or 8 Ω
Air gap:	25 µm
Transformer loss at 62 W:	0.2 dB
Weight:	2.5 kgs

* all windings connected in series

transformer as shown in Figure 4a.

The connections and circuit board layout for the primary are shown in **Figure 4c**. Simply slip the circuit board over the leads and solder it in place. The connections are marked as follows: supply voltage = Tr+, anodes = A / A*, screen grids = G / G*. Here '*' indicates the start of the winding.

In the original Philips design, the taps for he screen grids were at the 40-percent points of the windings, as measured from the centre tap. Here the proportion is 50 percent, which shifts the output stage more toward triode operation and causes the output power to be somewhat lower. In order to keep the coupling between the anode winding and the screen grid part the winding as tight as possible, windings on the same leg of the transformer have been matched together.

The transformer has eight secondary windings, which can be connected together in series or parallel in various ways in order to obtain the desired secondary impedance for the loudspeaker (4 Ω or 8 Ω) and the required primary impedance (6.0 kΩ). In the 4 Ω configuration, two sets of secondary windings are connected in series, while three sets are connected in series in the 8 Ω version.

The circuit board layout and connections for a 4 Ω loudspeaker connection are shown in **Figure 4d** (note the two wire links on the bottom side of the board, marked with short lines). **Figure 4e** shows the circuit board layout and connections for an 8 Ω loudspeaker impedance. In this case, there is only one wire link. Both configurations include a 1 kΩ shunt resistor at the output (R30). This resistor provides a certain amount of protection for the output transformer if no loudspeaker is connected. It also improves the stability of the amplifier with a capacitive load, such as may be present with a long speaker cable.

The leads for the secondary of the transformer are formed by bringing the tinned ends of the windings out to the terminal board. If you use one of the illustrated printed circuit boards for the 4 Ω or 8 Ω connections, bend the secondary leads flat against the wide tracks on the board and solder them in place.

The construction of the amplifier will be described in the next instalment. Since this involves a fair number of illustrations, a few plots of the measured performance of the amplifier are included in this instalment (see 'Performance').

(020071-1)

References

www.lundahl.se
– amplifier_30wpp.pdf
– appendix_cb.pdf
www.amplimo.nl

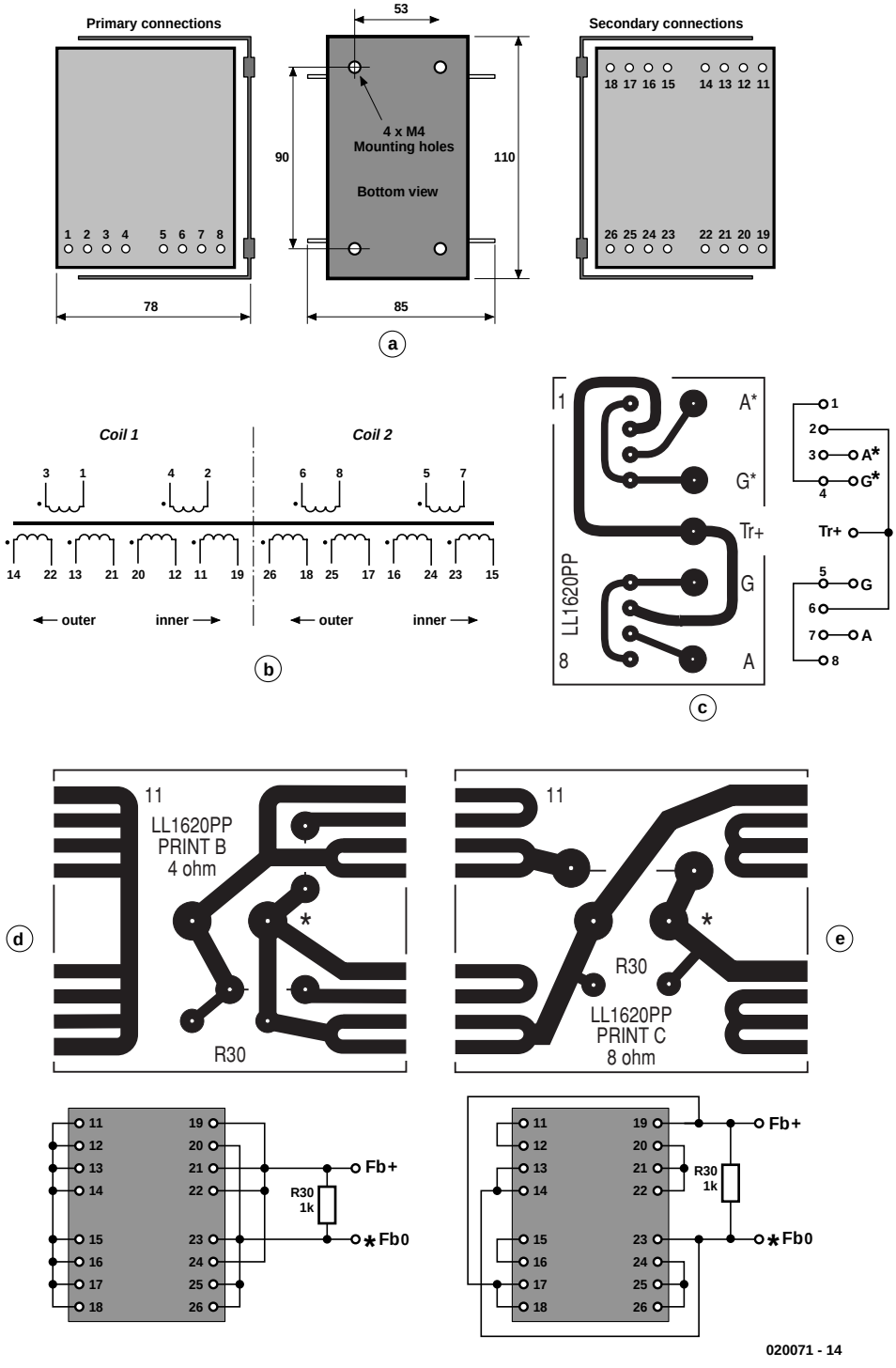


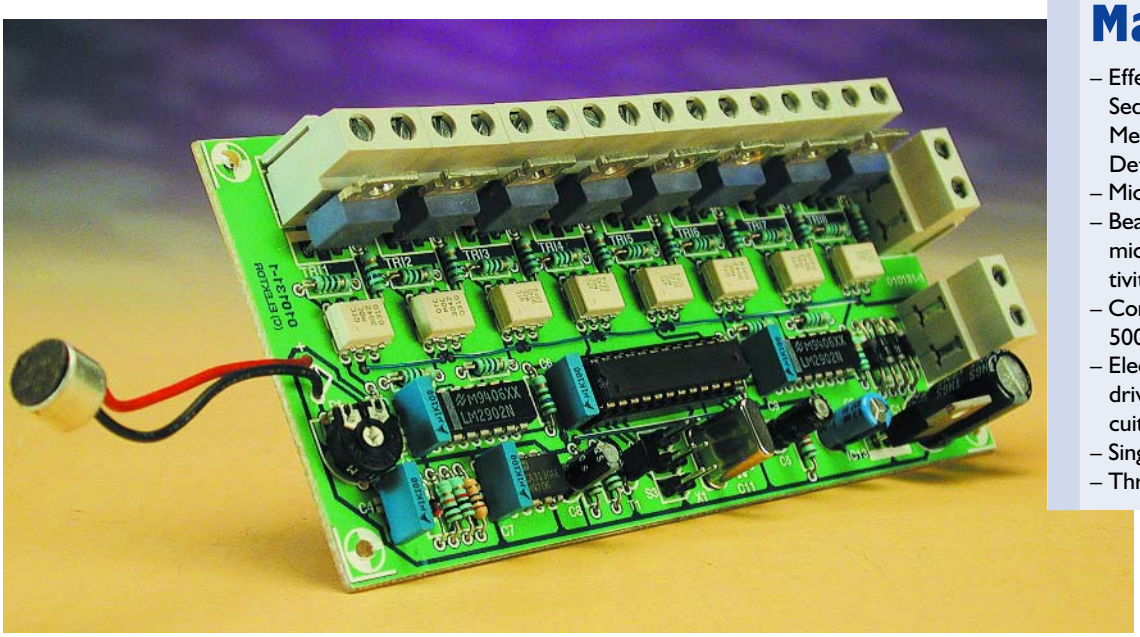
Figure 4. LL1620PP transformer: (a) dimensions and leads, (b) winding diagram, (c) primary winding connections and layout of the optional printed circuit, (d &, e) secondary winding connections and layouts of the optional printed circuit boards for 4-Ω (d) and 8-Ω (e) loudspeakers

8-Channel Disco Light Controller

up to 500 watts per channel

Design by Tom Varley

The unit is designed as a stand alone disco lighting controller but can be used for driving eight sets of Christmas tree lights, shop display lights etc. It is sound activated and self-running. The effects are blackout, flood, VU meter, streak and pattern sequence. The only controls needed are three pushbuttons. It's based around the popular 87C750 OTP device from Philips and was designed with a DJ or small band with a need to drive eight 500-watt PAR-56 lighting cans with a minimum of fuss.



Main features

- Effects: Flood, Blackout, Pattern Sequence, Beat Detect, VU Meter, Fast Chase on Beat Detect
- Microprocessor controlled
- Beat detector with built-in microphone and adjustable sensitivity
- Controls 8 stage lights of up to 500 watts each
- Electrical isolation between lamp drivers and microcontroller circuit
- Single-board construction
- Three-button control

This project should satisfy, at least for a while, those of you who have written to us clamouring for disco light effects units. Apparently commercial units with serious specifications are not only few and far between but also relatively expensive, hence our hopefully successful go at publishing a DIY unit. Don't be put off by 'hopefully', it's false modesty. As with all projects supported by printed circuit boards and software supplied through our Readers Services, the present project has been thoroughly tested by our in-house design staff, in close co-operation with the author,

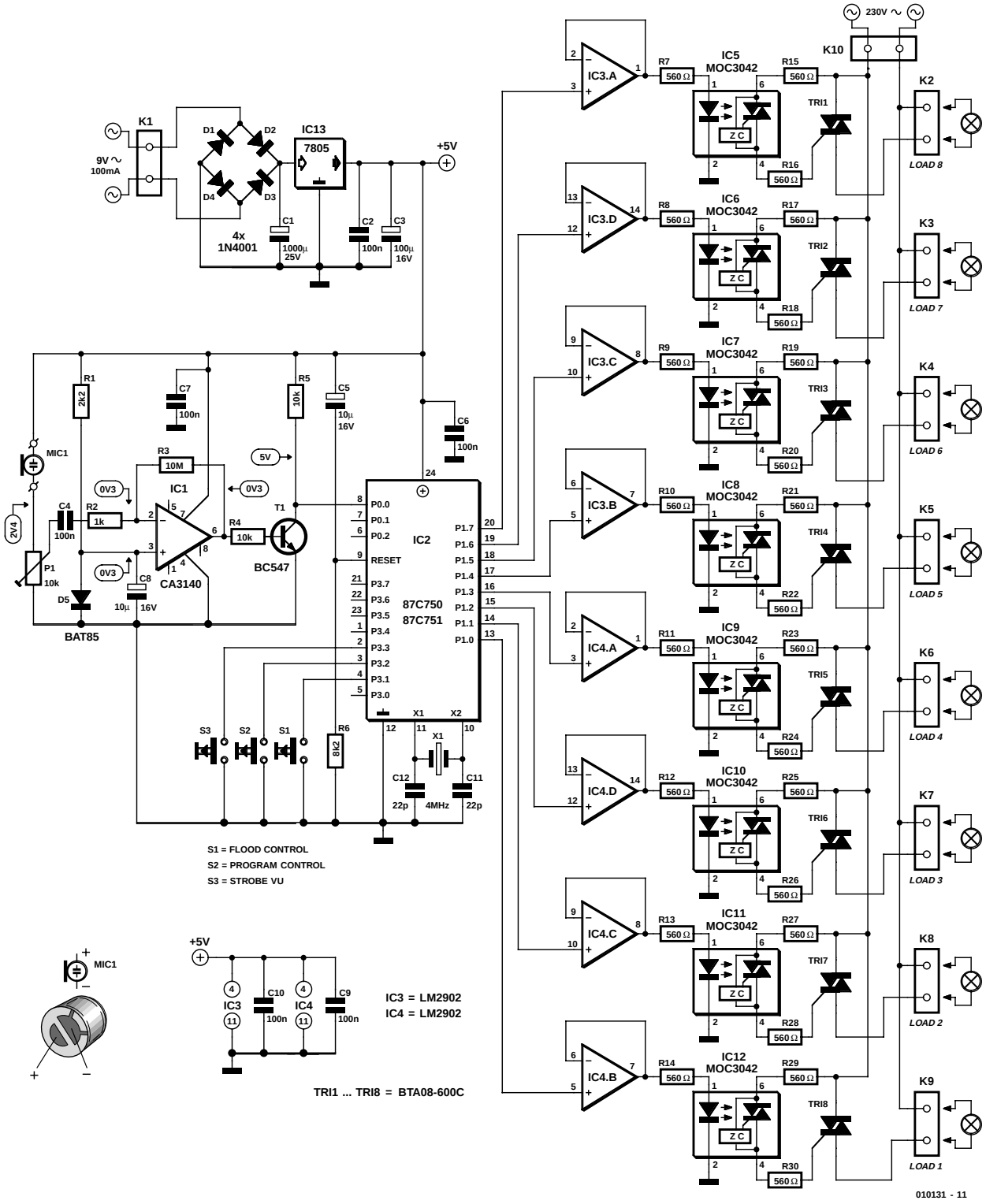


Figure 1. Circuit diagram of the 8-channel disco lights controller.

Tom Varley, who has live-tested the circuit with a performing band.

Circuit description

The schematic shown in **Figure 1** combines the digital and analogue sections in a single diagram. This is intentional because in reality these sub-circuits are contained on a single, single-sided printed circuit board.

The heart of the design is a microcontroller type 87C750 or 87C751, which has been programmed to look at the state of control switches S1, S2 and S3, as well as the output of comparator IC1.

Since the circuit is required to respond to acoustic stimuli (pounding bass speakers, etc.) a preamplifier is required to tell the microcontroller whenever such a signal is present. For this purpose a standard miniature electret microphone capsule with a DC

impedance of 2 kΩ is used. The microphone sensitivity is adjusted using preset P1 whose DC path also provides the capsule bias voltage. The gain of the preamp can be a few thousand times because neither distortion nor high-frequency response are an issue here. Because the opamp will only amplify alternating signals, its dc gain is unity (1?). Consequently the opamp inputs, and the output, are at a dc level equal to the forward drop of Schottky diode D5, or about 0.3 V. Using a 'normal' silicon diode in this position would produce a level of about 0.6 V which is enough to make T1 conduct all the time, and that is undesirable. When no sound is detected, the collector of T1 should be at about +5 V. Summarizing, an acoustic signal of sufficient level picked up by the microphone is turned into a square wave with a logic swing that can be detected and processed by the microcontroller, using its port line P0.0.

Almost any type of pushbutton switch can be used as there is a comprehensive debounce algorithm as part of the software running inside the micro.

Ports lines P1.0 through P1.7 of the micro drive opamp buffers type LM2902 (IC3-IC4). Each of the buffered opamp outputs in turn drives an LED inside an opto triac driver (IC5-IC12). The micro needs buffering to reduce load constraints on the port lines. The opto-triac drivers are a simple method of providing zero crossing (ZC) to reduce switching noise while providing a few kV of electrical isolation between the 'hot' lamp circuits and the delicate electronics. The triacs Tri1-Tri8 may be wired via PCB terminal blocks to two Bulgin sockets or eight individual mains sockets for connection of the lights or light units. More about this further on.

The author experimented with different values for biasing the triac driver. In practice, 560 Ω works fine and being the same value as the LED bias resistor reduces the number of resistor values. The MOC3042 may be replaced by its slightly more sensitive sibling the MOC3043. The MOC3041 is not suitable here. The triacs, too, are uncritical, and virtually anything may be used that's capable of switching 400 V at about 4 A.

The standard power supply arrangement of a ready-made mains adapter (9 VAC or 9 VDC, 300 mA) or a dedicated transformer, a bridge rectifier (D1-D4) and a fixed voltage regulator (IC13) is a good low-pass filter for mains-borne noise and transients. The supply should always include the 0.1 μF decoupling capacitor. Thanks to the bridge rectifier, the supply works irrespective of the output polarity of the mains adapter.

The microcontroller is running at 4 MHz using a quartz crystal, X1, and the usual pair of small ceramic capacitors, C11-C12. C5 and R6 are the boot-up components that hold the RESET pin high for a minimum of two machine cycles when power is applied to the board. This is the recommended cold start for 8051 architecture devices.

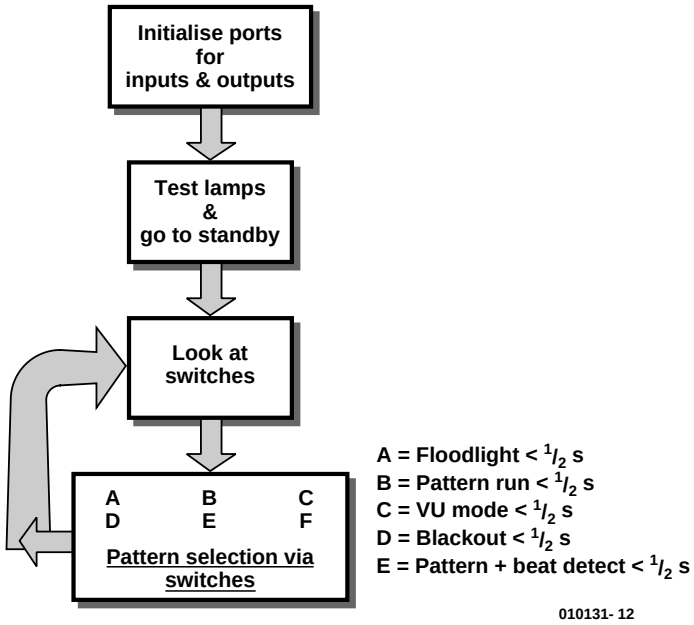


Figure 2. Flowchart of the pushbutton reading routine.

Software and operation

The program inside the 87C750 (or '751) black box was written as assembler and is about 150 lines of code, the rest of the program memory area (1 K for the 87C750) is used as a lookup table which simply determines which lamps are turned on. After the board is powered up the micro is initialised so that port 1 is briefly strobed, driving all the lights on in sequence, to give visual indication of power up. After the strobe is run, the device puts itself into a loop and starts to look at the switches. Each switch has two functions. The first function is achieved

COMPONENTS LIST

Resistors:
R1 = 2kΩ2
R2 = 1kΩ
R3 = 10MΩ
R4,R5 = 10kΩ
R6 = 8kΩ2
R7-R30 = 560Ω
P1 = 10kΩ preset

Capacitors:
C1 = 1000μF 25V radial
C2,C4,C10 = 100nF
C6,C7,C9 = 100nF, 7.5mm lead pitch
C3 = 100μF 16V radial
C5, C8 = 10μF 16V radial
C11,C12 = 22pF ceramic

Semiconductors:
D1-D4 = 1N4001
D5 = BAT43 or BAT85 (Schottky)
IC1 = CA3140, CA3130
IC2 = 87C750 or 87C751, programmed, order code **010131-41**
IC3,IC4 = LM2902
IC5-IC12 = MOC3042
IC13 = 7805
T1 = BC547

Miscellaneous:
K1-K10 = 2-way PCB terminal block, lead pitch 7.5mm
MIC1 = electret capsule
S1,S2,S3 = pushbutton, 1 make contact
TRI1-TRI8 = BTA08-600C (BCR6 AM 8)
X1 = 4MHz quartz crystal
PCB, order code **010131-1**

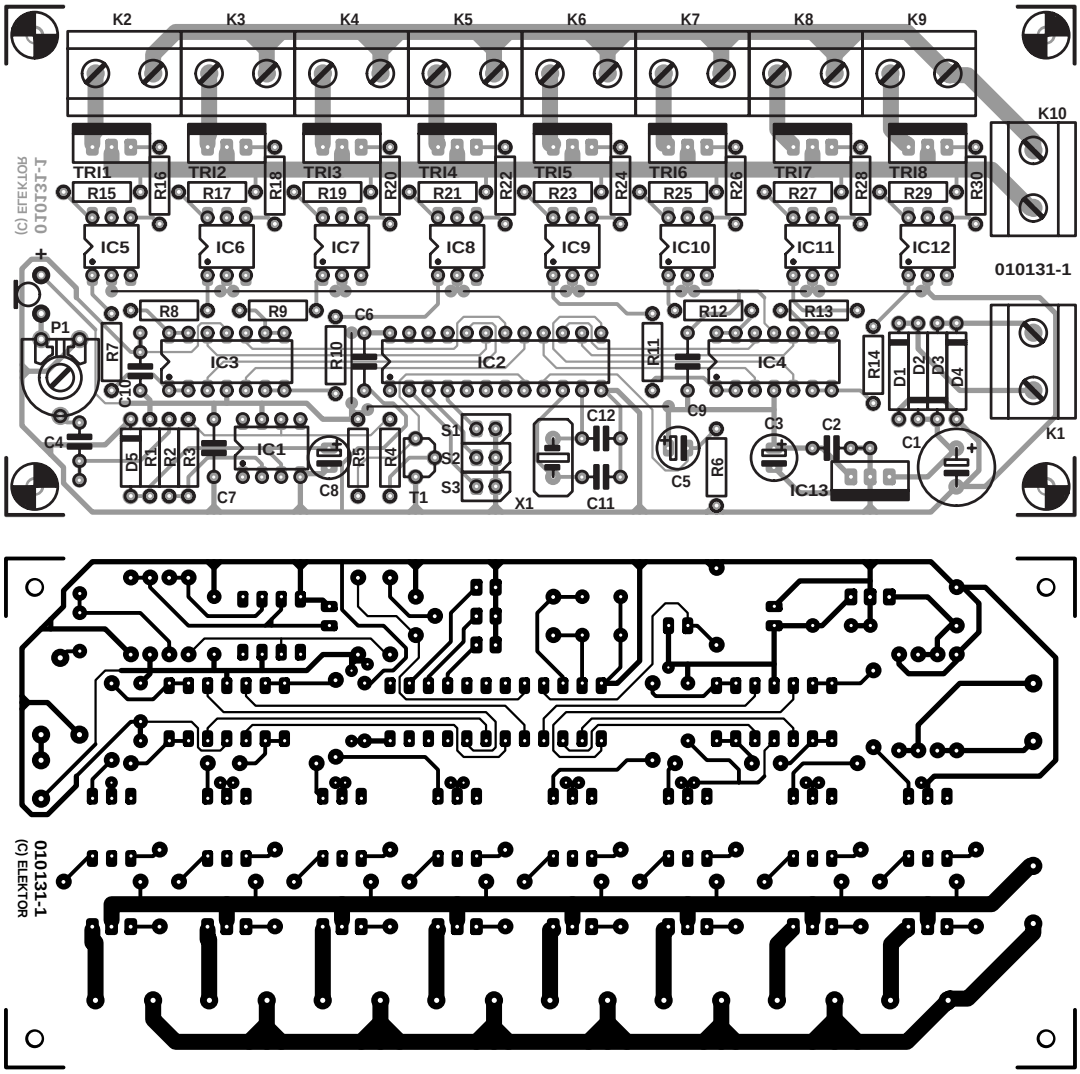


Figure 3. Copper track layout and component overlay of the single-sided PCB designed for the unit (board available ready-made).

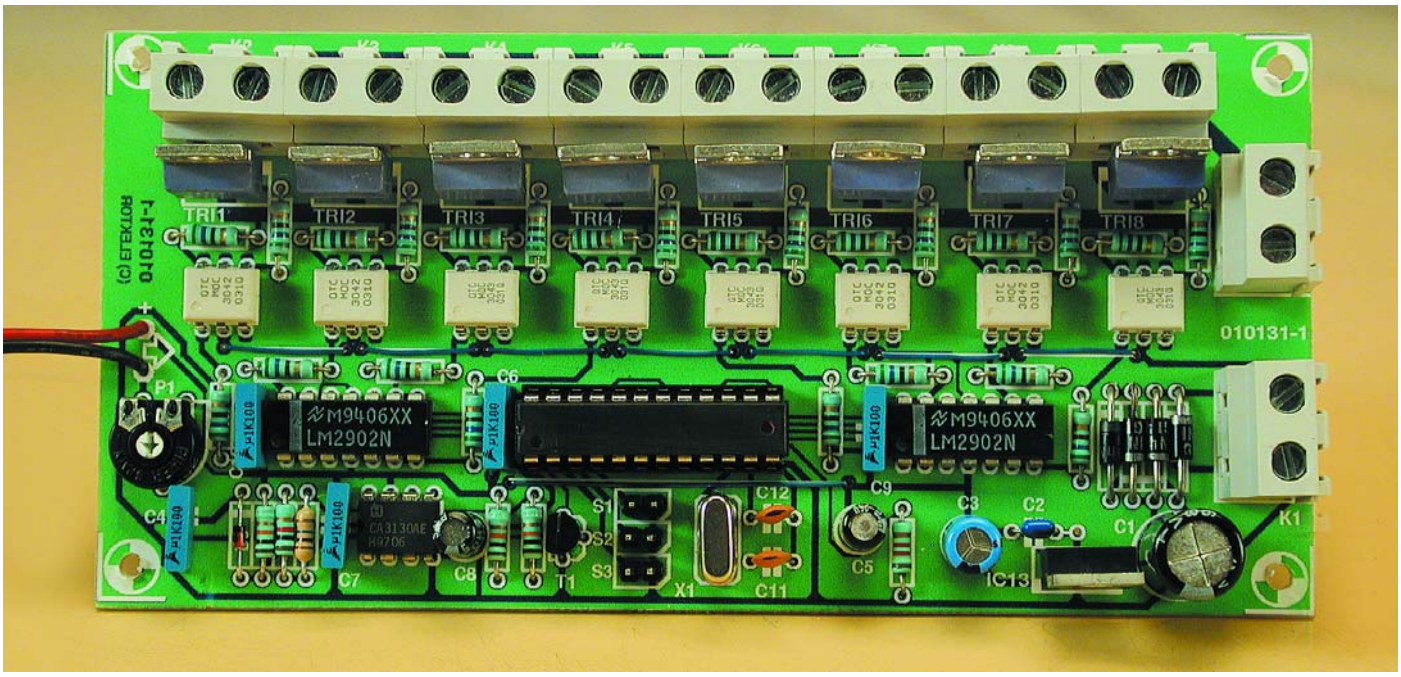


Figure 4. This is what our finished an tested board looks like. Note that the three pushbuttons (not shown) are connected via pin strips.

by depressing the switch for less than half a second, these functions are A, B & C. The second layer of functions are reached by depressing the relevant switch for more than half a second. It should be noted that if you keep the switch depressed it will loop around the top and bottom functions of the switch. The chart in **Figure 2** indicates which function can be selected. Once selected, the controller will operate in this mode until a new function is selected.

Because the author receives royalty payments for every ready-programmed microcontroller sold, the assembly code listing of the program burned into IC2 will not be made available as usual through our website, or on a floppy disk. Ready-programmed ICs for this project are however available from Readers Services, the order code is **010131-41**.

Assembly

The electronics fits neatly on a **single-sided** PCB, the layout of which is shown in **Figure 3**. No problems or difficulties are envisaged if you work carefully, take your time and stick to the parts list and the component overlay printed on the board. For your convenience, our tested prototype is shown in **Figure 4**. There are **nine wire links** on the board, we suggest counting them to make sure you haven't missed one.

The electret capsule is connected to the PCB via a short two-wire cable. Depending on how well sealed the case is and how flexible the case walls are will determine the bass sensitivity of the microphone. If air can pass through the case with ease then the microphone will be sensitive to bass. Experimentation with the mechanical coupling of the microphone to the PCB or case wall will also change the acoustic response characteristics. The choice of the electret capsule is not critical, and even the cheapest ones will work fine. Usually, the capsule has two solder pads at its rear side. The pad connected to the metal encapsulation is the negative (–) connection (see also the insert in the circuit diagram).

The triac output connectors K1-K8 may be wired to 8 separate mains sockets allowing each stage light (or cluster of lights) to be powered via its own cable.

Some of you may want to employ two 'Bulgin' 8-way sockets, one for each group of four lamps. The easiest way of wiring to the IEC socket and the Bulgin connectors is to prewire tails from the PCB pins and thread these tails through the relevant holes in your case. Once the PCB is secured inside the case you can then solder the sockets. If you are using a steel case make sure that all parts of the case are earthed to the IEC socket earth pin.

If used, the Bulgin connectors should be wired as follows:

Pin 1	Protective Earth
Pin 2	Light 4 (or 8)
Pin 3	Light 3 (or 7)
Pin 4	Light 2 (or 6)
Pin 5	Light 1 (or 5)
Pin 6	n.c.
Pin 7	Neutral
Pin 8	Neutral

Whatever the method used to connect the stage lights to the circuit, **great care must be taken and proper materials used to eliminate any risk of the mains voltage being touched**. In this respect, we advise (re-)reading our notes on Electrical Safety which are published every few months in this magazine.

Testing

Warning

The mains voltage must be removed from K10 before doing any measurements or testing on the circuit. When the mains voltage is present on K10, the tabs of the triacs are LIVE. No matter if you decide to house the circuit in a solid plastic (ABS) or a steel enclosure, a proper connection to the mains Protective Earth (PE) terminal must be provided. The isolation distance on the circuit board is smaller than 6 mm so this equipment meets Class 1 only.

With no lights connected and the microcontroller out of its socket, power up via the mains adapter. Check for 5 volts at the regulator output, if it's not there check the diodes D1-D4 are the right way round and are being fed with the proper AC or DC voltage.

Once you have 5 volts, you can set up the microphone sensitivity. The room needs to be free of excessive background noise. Using a small screwdriver, rotate P1 fully counter-clockwise. Connect a scope probe or the DVM meter leads to IC1 pin 6. Slowly rotate the wiper and stop when pin 6 goes Low. Rotate the wiper counter-clockwise again and stop when the output swings High. Now clap your hands or whistle and the output should briefly change state when the microphone picks up the sound. Carefully tweak the setting of P1 until the toggling takes place reliably and at the desired acoustic level. Do remember that ambient noise levels will be very high in a discotheque!

Power down and install the microcontroller. Make sure the PCB is securely installed in the case and that the stage lights are connected to PCB terminal blocks K1-K8. If you do not have a set of lights handy you can either wire up 8 sets of fairy lights or hardwire 8 batten holders. Close the case and connect the mains power to K10.

Plug in the mains adapter and test the switch functions. Switch 1 if depressed will continually cycle with the light pattern (binary) as follows :

```
*******
*- - - - - - *
 - - - - - - - -
```

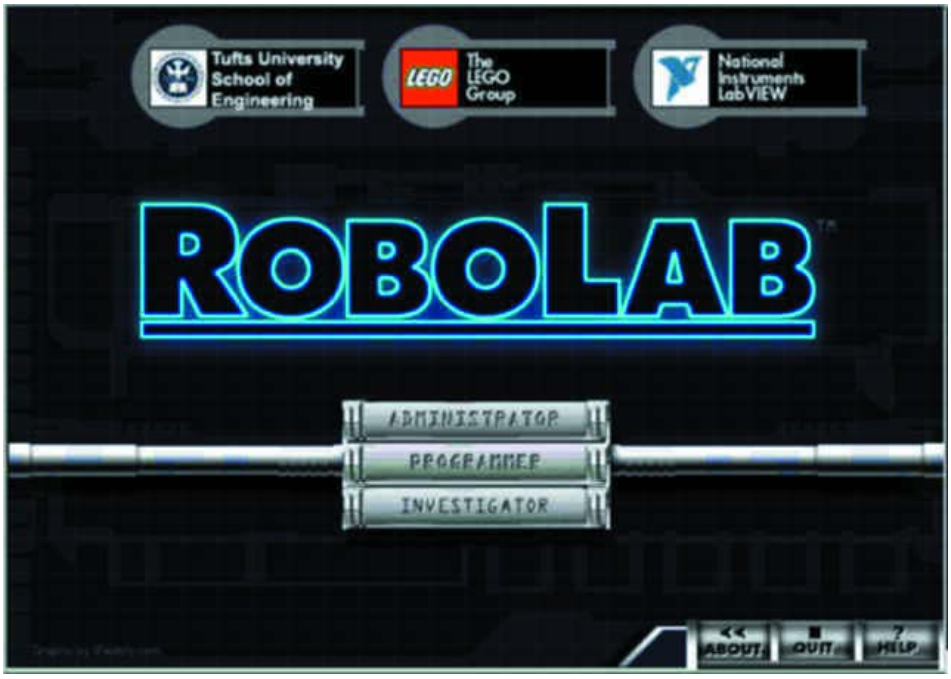
(010131-1)

RoboLab

Mindstorms five years on

By C. Bauman

Back in 1998 the Danish toy manufacturer LEGO introduced its radical MINDSTORMS kit featuring the RCX programmable brick. Much effort and expense had been invested in its development and success was by no means certain. As it turned out the kit was well received and the concept was so well thought-out that research and educational establishments began buying them as well.



When it comes to choosing high-tech computer games for children, today's parents are often put off by their negative aspects and see little educational worth in achieving, say, the next level up in a role playing game (RPG, see *Elektor Electronics* January 2003). Maybe they are nostalgic for a time when they themselves built things together with their own parents using Meccano or spent hours tinkering with a train set. In 1998 when LEGO introduced Mindstorms we suddenly had a toy that could both entertain and educate. The slogan that LEGO used at the time was *'simple programming — even for adults'* and in fact this was literally true for many parents who were introduced to the possibilities of robotics and software for the very first time.

Since its introduction the RCX has also found its way into some of the more progressive primary and secondary schools where its worth as a teaching tool, as originally conceived by Prof. Seymour Papert (M.I.T.) has been recognised and exploited.

The programmable brick has not stopped there, a quick trawl on the internet will produce enough evidence to show that there is hardly a technical college or university (mostly in the US) that does not have some area of its web site dedicated to RCX robotic projects. Some of the more notable examples can be found at: **Stanford University [1]**, **Loughborough University [2]** and **Edinburgh University [3]**.

The challenge with Mindstorms is to produce ever more sophisticated robots capable of performing more complex tasks. The possibilities are increased by interfacing new sensors to the RCX (see *Elektor Electronics* May 2002) and using CAD programs to assist in construction and of course by clever programming. Just looking at the plethora of alternative software/firmware available for the RCX and browsing through some of the books available with titles such as *Extreme Mindstorms*, *Ultimate Tool*, *Definitive Guide* gives some insight into the

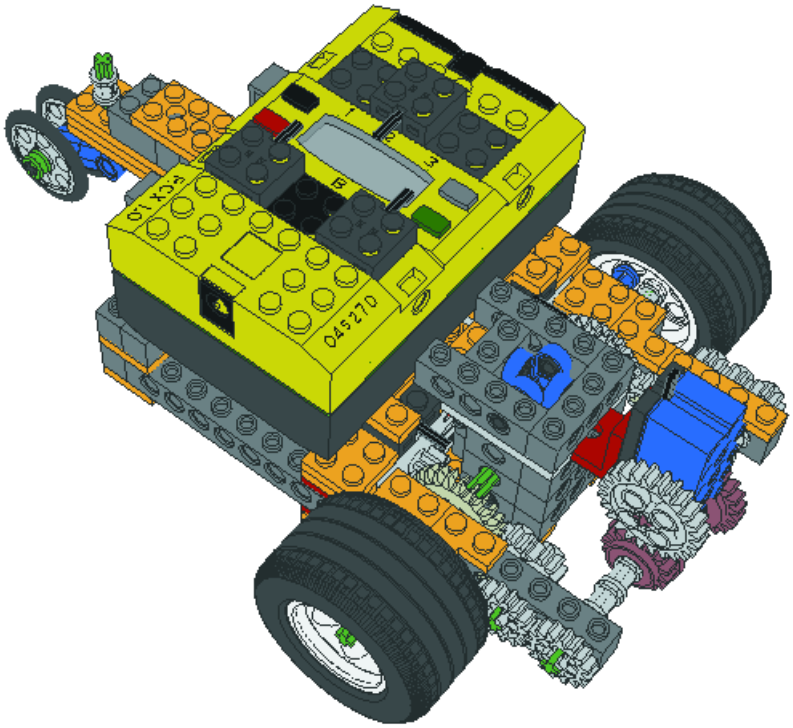


Figure 1. A close-up of Roby.

level of expertise and support that has grown up around this system.

Those icons

ROBOLAB uses a series of colourful, descriptive and expression-strong icons that can easily be learnt and are well documented. The difference between source and destination are well defined. Diamond shaped icons are called modifiers in ROBOLAB. Source modifiers have blue data connecting paths and a blue frame. Destination modifiers have a green or brown frame with the same colour data paths. In LabVIEW the different data types are represented by different coloured and sized threads.

The RCX programming environment has also seen an improvement with the release of the second edition of its firmware allowing the use of: *Events*, *Access-control* and *Local Variables* for example. All these features of Version 2 of the RCX operating system cannot be used by Visual BASIC because a new version of *Spirit.OCX* has not yet been released. C++ however can use the new features via **SDK2 [4]** for example. The popular *NQC* (Not Quite C) written by **Dave Baum [5]** has access to most of the new features.

In parallel to these developments enthusiasts have also been hard at work writing alternative operating systems for the RCX brick. Two of the most popular open-source operating systems are *legOS* (renamed *BrickOS* after pressure from LEGO) using syntax similar to NQC and *lejOS* based on Java. Fans of the computer language Forth may be interested in *pbforth* from Ralph Hempel.

The RCX is also a popular choice for design prototyping in many Universities. In these specialist environments the limitations of the standard firmware can sometimes be restricting but it is not always necessary to resort to third-party software to unlock the potential of the RCX. Prof. Chris Rogers and his co-workers at **Tufts University Massachusetts [6]** together with the company LEGO and National Instruments (NI) have developed the ROBOLAB programming environment.

ROBOLAB is an icon based graphical programming environment based on LabVIEW. It supports all the phases of robot design from the original concept and planning, programming and trials through evaluation, documentation and finally publishing the design. The ROBOLAB environment was originally conceived as a teaching tool for schools but it has found acceptance by all age ranges from primary up to University level.

The ROBOLAB software has picked up numerous awards worldwide including the 2001 Eddy award (the annual Apple Mac award for top hardware/software products) and the BETT prize awarded by members of the British education and teaching professions.

ROBOLAB — currently released as version 2.5 — is a versatile, advanced programming environment with enough features to allow implementation of the most sophisticated programming tasks. Some of its more powerful features are:

– Programming one RCX from another RCX.

– Programming and data logging via the Internet.

– Combining a LEGO camera together with the RCX at the highest levels.

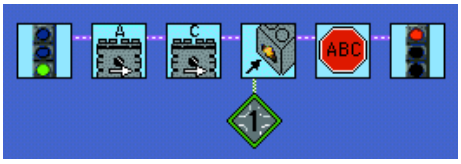
– Data evaluation using LabVIEW Gcode.

If you are in any doubt about these claims checkout **[7]** where the author describes in detail the development of a robot arm project. The arm has five axes of movement (degrees of freedom) with a reach of approximately 50 cm. Programs have been written to enable the arm (together with a LEGO camera) to identify and collect LEGO bricks and place them in a box (wow, self-tidying toys!) and also to build a four-storey tower from LEGO bricks. The programs were written using only the ROBOLAB environment.

onward!

To demonstrate the main features of ROBOLAB a small three-wheeled autonomous roving robot called **Roby [8]** was built. Roby has two motors driving the left and right wheels and a touch sensor mounted at the front (initially both left and right touch sensors are

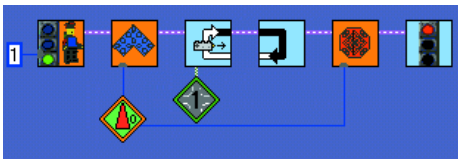
connected in parallel to port 1). In this simple example the robot will be programmed to go straight ahead until its sensors detect an obstacle in its path.



Listing 1. Every ROBOLAB program starts with a 'begin' icon and finishes with an 'end' icon. The motor starts and then runs until the touch sensor is triggered.

In ROBOLAB just as in LabVIEW data is passed using virtual 'threads'. In this program there is only one main thread directly controlling the motors. There is only one task. After initialisation the program remains in a loop constantly monitoring the touch sensor, as soon as a contact is detected the motors will be switched off.

The same program function can be re-written in a slightly different way.



Listing 2. At the start of the arbitrator program the request is given a priority of 0 to drive straight ahead if the touch sensor is not triggered; request the drive to stop when a sensor is triggered.

The first icon in the program sequence is not exclusively for initialisation but also indicates a background task is started to control the motors (resources). It functions as an arbitrator to decide which of the following tasks will be allowed to control the motors. The individual threads (in this case only one) are assigned a priority (0 = highest, 4 = lowest Priority) so that the arbitrator constantly monitors requests and always allows the highest priority task to have control over the motors. In this simple example it is difficult to see the benefits of this architecture but as the robot control gets more sophisticated with more complex tasks competing for control of the resources then the advantages become more obvious.

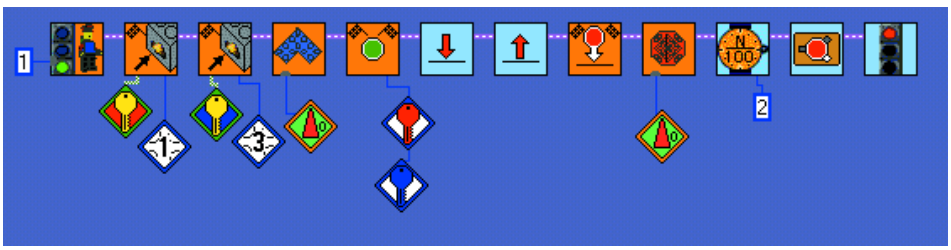
This particular type of robot programming architecture was developed in the late 80's by Rodney Brooks at M.I.T. and is called **Subsumption Architecture [9]**. Traditional multi-tasking environments perform poorly if several tasks attempt to control a common resource with conflicting commands. In contrast the arbitrator in subsumption architecture ensures that the highest priority task only has control.

The software response time is quite slow but we can improve this by using interrupts. Version 2.0 of the LEGO firmware allows interrupts to the RCX processor (a 16 MHz H8/3292 from Hitachi) to be used in the software. With a few modifications to the program we can now use interrupts to improve the reaction time. The modified program is shown in **Listing 3**.
The two touch sensors icons following the arbitrator icon indicate that triggering either of the two touch sensors (now connected independently to ports 1 and 3) generates an interrupt represented by the red and blue keys under the icons.

The fifth icon enables the interrupt and is preceded by a request to drive straight ahead. When a touch sensor is activated the program jumps out of the continuous loop to the last part where it executes the stop command, waits 2/100 s and then ends all the tasks including the arbitrator. In the previous example the program did not terminate.

This example touches on some of the simpler features of ROBOLAB, to get a better impression of the software, a function-limited demo version of can be downloaded from **[10]**.

(020319-1)



Listing 3. The same program but this time using an interrupt to speed up the reaction time.

Stanford University [1]
www.stanford.edu/class/ee380/9899fall/lect03.html
Loughborough University [2]
www.lboro.ac.uk/departments/el/robotics

Edinburgh University [3]
www.dai.ed.ac.uk/groups/mrg/research/LEGO_project.html

SDK2 [4]
www.ceeo.tufts.edu/robolabatceeo/Resources/documentation/LEGOMindStormsSDK.zip

Dave Baum [5]
www.enteract.com/~dbaum/nqc/index.html

Tufts University Massachusetts [6]
www.ceeo.tufts.edu/robolabatceeo/default.asp

Robot arm Project [7]
www.convict.lu/Jeunes/5%20DOF%20Robot-arm.htm

build details for Roby [8]
http://www.convict.lu/Jeunes/LOGI2.htm

Subsumption Architecture [9]
www.convict.lu/Jeunes/Subsumption.htm

ROBOLAB demo [10]
www.technik-lpe.de/lego_dacta/li_down/ROBOLAB_DEMO_20.zip

Practical Neural Networks (4)

Part 4 (final): applications and large neural nets

By Chris MacLeod and Grant Maxwell

In this final part of the series, we will consider some of the other applications of neural nets and have a look at a few of the more difficult questions facing neural net researchers.

During the series we've concentrated mostly on image recognition. This is because it is not only one of the most important applications of neural nets, but also the one that's easy to illustrate, as we can "see" the images.

However, as explained in Part 1, the neural net can be considered as a universal logic system, capable (providing that the network has three layers) of learning to produce any truth table you like, see **Figure 1**.

If the output of the neurons is a sigmoid function, then it acts rather like 'fuzzy logic', and is able to produce analogue outputs — which can be useful for handling problems, in the real world, which are not 'black and white'.

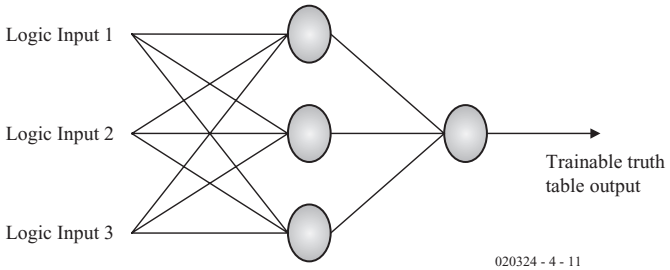


Figure 1. Universal trainable logic.

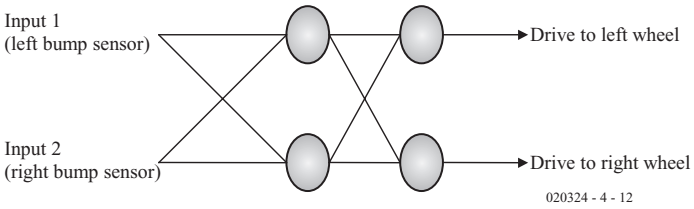


Figure 2. Robot controller.

Applications in Robotics

To illustrate how a trainable truth table is useful, consider a robot controller as shown in **Figure 2**.

The network can learn to make the robot behave as we wish if we train it to respond to the sensor inputs and produce outputs to control its motors. This is a simple example of what is termed an Artificial Nervous System (ANS) and an animal-like robot controlled through a learning control system is sometimes referred to as an Animat.

You could teach this robot to behave as you wish with the Back Propagation Algorithm. Alternatively, you could have it learn to survive by itself, using a **re-enforcement** learning algorithm. Re-enforcement methods work by rewarding behaviour which benefits the robot (e.g., avoiding obstacles) and strengthening the weights which caused this behaviour. This can be accomplished automatically in the robot itself (of course we need to make some judgements about what's good and bad behaviour and program these in, so that the robot has a **value system**).

A slight variant on this is to use a Genetic Algorithm to evolve network types from a population which lead to good behaviour — this algorithm is called Evolutionary Algorithm for Re-enforcement Learning (EARL).

Applications in Control

We can extend the control aspects mentioned above to other systems. For example, suppose we want a system to control a DC motor, see **Figure 3**.

You might recognise the inputs as the typical PID types used in

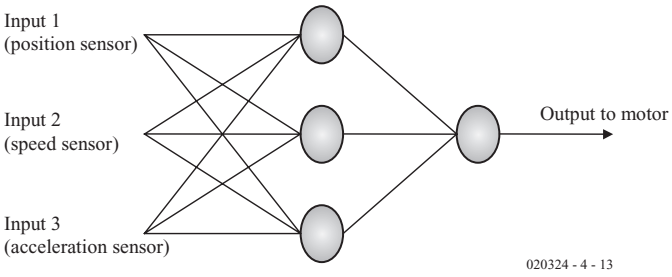


Figure 3. Motor control.

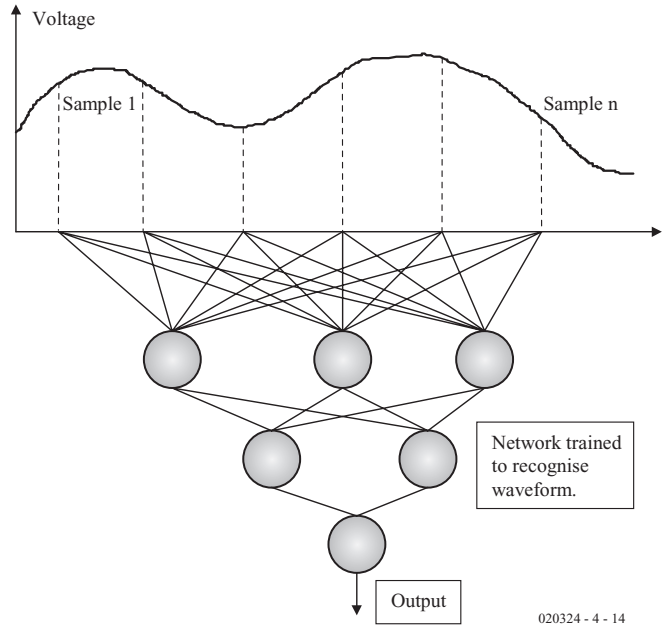


Figure 4. A sampled waveform fed to a neural net.

conventional controllers. A **neuro-controller** like this might learn with BP, from a conventional controller, by using its outputs as targets.

Alternatively, and more interestingly, it could learn from experience, using a Re-enforcement Learning Algorithm or a Genetic Algorithm to set its weights.

Man Machine Interface and Intelligent Sensing

In the control system above, the network responded to sensors and produced an appropriate output. Neural Nets are good at untangling complex interrelated inputs like this — you don't have to understand the theory of the system, just have examples to train the network with. The ANN will sort itself out during the training process to make sense of the relationships and transformations between inputs and outputs.

One future application for this type of 'intelligent sensing' system is a human to machine interface. If you were to try, for example, and make realistic prosthetic limbs and interface them with part of the nervous system, you'd have to try and make sense of the outputs of the many nerves (and perhaps other biometric inputs and noise), all firing with different frequencies. Some of these inputs may be irrelevant and others related to each other in subtle ways. Processing information from this type of complex and non-linear system is obviously something which is right up the neural net's 'street'.

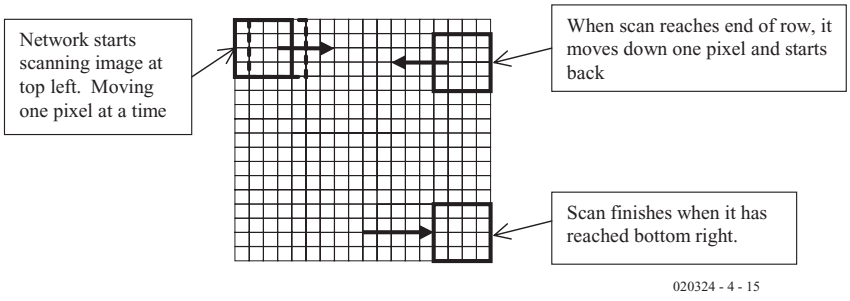


Figure 5. Image scanning.

Recognising Waveforms

In the man-machine interface discussed above, you'd have to process the waveforms generated by nerves. And indeed, the Neural Net can recognise patterns in time (like these) as well as in space (like images), by sampling them and feeding them to the inputs; **Figure 4** shows the idea.

Of course this leads to the problem, already mentioned in part 2, of the image or, in this case, waveform, having to be exactly the right size and being placed in the middle of the grid.

Pre-processing

Sizing and centring an image or data, so that it's suitable for a network, is called pre-processing. There are many different ways of achieving it. In the case of a sampled waveform, the network is looking at a 'window' of time, and as the waveform passes it by, one sample at a time, it will eventually lie in the centre of the window and the network will recognise it.

Exactly the same thing can be said of an image-recognising network. It to can also be scanned across the image, as shown in **Figure 5**. Again, during this process, the pattern to be recognised will eventually end up in the centre of the grid. Actually this is not dissimilar to what our eyes do when we study a scene, we (unconsciously) scan the image, looking for patterns to recognise. One important point about training a network for this sort of task, is that we have to train it to recognise 'noise' or irrelevant data as well as the wanted pattern or it will give false positives.

So much for centring the image; what about making it the correct size? Well, again this can be done by pre-processing. If we start with a large window and scan it across

the image, we can make the window progressively smaller, so that sooner or later the pattern we want to recognise is correctly sized on the grid, see **Figure 6**.

Since the neural net inputs are of a fixed size, fitting different sizes of image into it means manipulating the original image size until it fits onto the grid. This is easily done by averaging adjacent pixels together until the image is the correct size for the network.

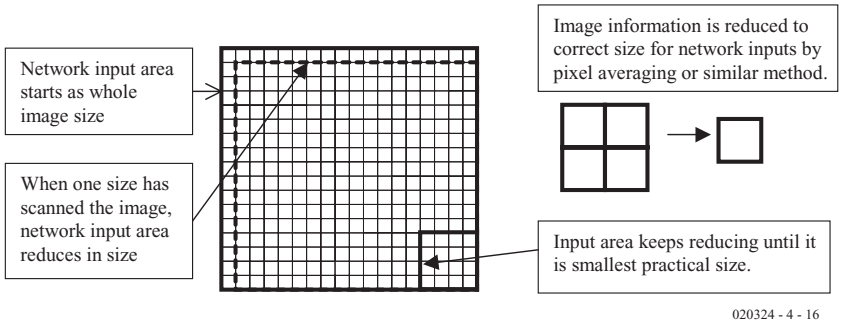


Figure 6. Image sizing.

Modular Networks

The neural networks in the brain recognise objects through features. Take the letter L, for example. It has two features, a vertical line and a horizontal line. No matter where it is placed on a grid, nor how small or larger it is, it still has these two features.

If we could have a network which recognises verticals and another which recognises horizontals, then a third can integrate them together to form a whole — have a look at **Figure 7**.

This is a simple example of a modular network. Rather than one large and fully connected network trying to learn everything, we break the task down into smaller chunks, each one tackled by a smaller net. Research has shown that the brain is organised in this way — not as one huge net, but as lots of smaller modules acting on their own tasks and integrating their results together into a coherent whole.

Research is ongoing at the moment to see if it is possible to use Evolutionary Algorithms to grow groups of such modules and form an integrated system.

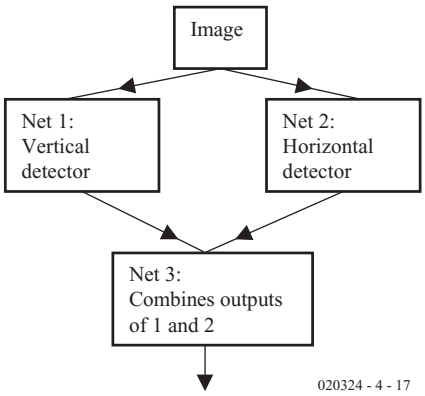


Figure 7. Modular networks.

Neural Function

The sharp-eyed reader will have spotted that the description of the Artificial Neuron given in part 1 doesn't bear too much resemblance to the Biological one described. In particular, the biological neural net frequency-coded its activity, as shown in **Figure 8**.

Some scientists think that this is important and that, what we term 'thought' and 'consciousness' are in fact sequences of these pulses racing around the nervous system, interacting with one another. On a more pragmatic note, many control systems are easier to work with when information is coded like this (or pulse-width coded).

Artificial models have been made to operate in this manner and are often called 'spiky' neurons. **Figure 9** shows two simple models, one of a 'spiky' (frequency modulated) neuron and the other of a pulse-width modulated neuron. The weights and other parameters of these are easily set using a GA (or if you fancy a bit of mathematics, by modifying BP).

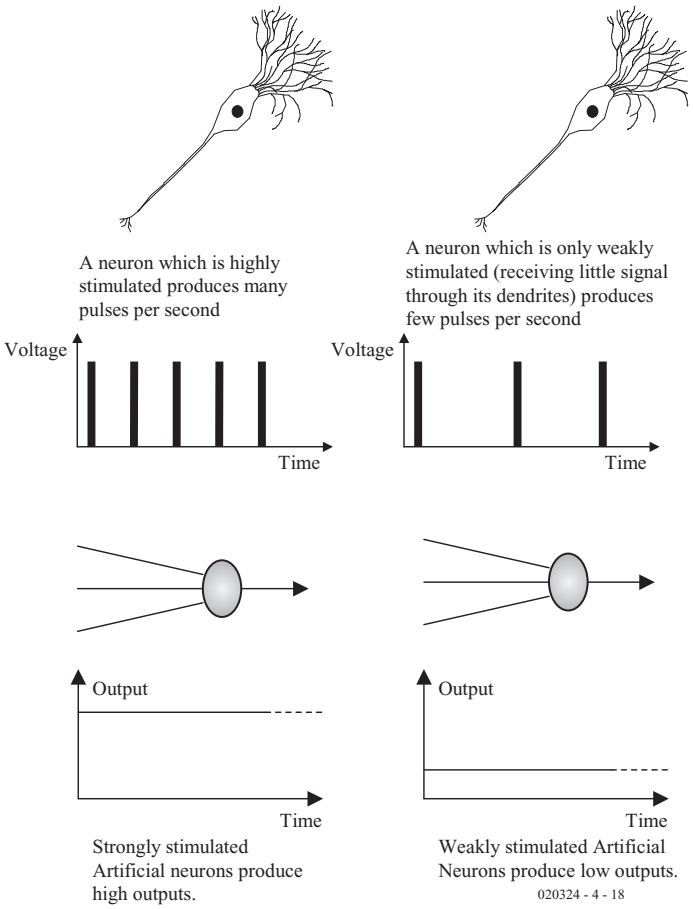


Figure 8. Activity in a biological and artificial neuron.

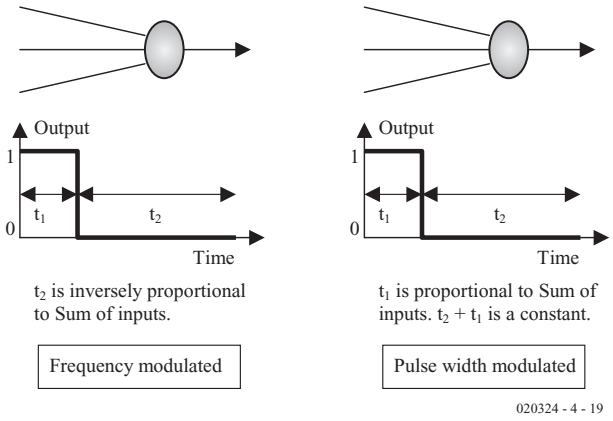


Figure 9. Spiky neurons.

Connections and technology

One final area of interest relates to the size of the biological neural net compared with the artificial one. You may remember that the brain is built from 100 billion neurons (the same number of stars as there are in our galaxy). This easily makes it the most complex structure in the known universe.

On top of this, each of these neurons may be connected with 1,000 others (some neurons have 100,000 connections). This means that there are perhaps 100 trillion connections in the brain.

Modern technology cannot match either the density of neurons or their connections. The neuron is actually about the same size as a transistor, but is arranged in a dense three dimensional structure. Of course, our circuits are essentially laid flat — two dimensional.

It may seem that both the density of neurons and their wiring are insurmountable problems if we wish to make biologically realistic neural nets. But, it should be remembered that signals travel through real neurons at only a couple of hundred metres a second, whereas in the artificial equivalent they can travel at sixty percent the speed of light. So, what the artificial net loses in connectivity, it gains in speed.

It might be possible to solve the connectivity problem, even with current technology, if each neuron is given an address like a host in a communications network and this is used for routing signals to and from neurons rather than having delegated wiring for each signal.

Making hardware neural nets is a problem as there are no integrated circuit technologies readily available for the amateur to experiment with (large discrete neural nets require many components) — particularly ones with reconfigurable wiring, which is required for Evolutionary ANNs. Some experimenters have tried Field Programmable Gate Arrays (FPGAs), but these are not ideally suited for the task as each neuron takes up a great deal of chip 'real estate'. This type of problem exists because the neural net was modelled on the brain without regard for electronics — and so is inefficient to implement. We also need to develop a neuron (in the sense of a general purpose processing unit, capable of learning) which is electronically feasible and efficient.

Correction

In part 2, some errors have crept into the text which explains Figure 4. Some of the equations were spread over two lines and the resulting hyphen is easily confused with the minus sign. Also, one of the equations contained a mistake. The correct equations are given below.

1. Calculate errors of output neurons

$$Error_\alpha = out_\alpha (1 - out_\alpha) (Target_\alpha - out_\alpha)$$
$$Error_\beta = out_\beta (1 - out_\beta) (Target_\beta - out_\beta)$$

2. Change output layer weights

$$W^+_{A\alpha} = W_{A\alpha} + \eta Error_\alpha out_A$$
$$W^+_{A\beta} = W_{A\beta} + \eta Error_\beta out_A$$
$$W^+_{B\alpha} = W_{B\alpha} + \eta Error_\alpha out_B$$
$$W^+_{B\beta} = W_{B\beta} + \eta Error_\beta out_B$$
$$W^+_{C\alpha} = W_{C\alpha} + \eta Error_\alpha out_C$$
$$W^+_{C\beta} = W_{C\beta} + \eta Error_\beta out_C$$

3. Calculate (back-propagate) hidden layer errors

$$Error_A = out_A (1 - out_A) (Error_\alpha W_{A\alpha} + Error_\beta W_{A\beta})$$
$$Error_B = out_B (1 - out_B) (Error_\alpha W_{B\alpha} + Error_\beta W_{B\beta})$$
$$Error_C = out_C (1 - out_C) (Error_\alpha W_{C\alpha} + Error_\beta W_{C\beta})$$

4. Change hidden layer weights

$$W^+_{\lambda A} = W_{\lambda A} + \eta Error_A in_\lambda$$
$$W^+_{\Omega A} = W_{\Omega A} + \eta Error_A in_\Omega$$
$$W^+_{\lambda B} = W_{\lambda B} + \eta Error_B in_\lambda$$
$$W^+_{\Omega B} = W_{\Omega B} + \eta Error_B in_\Omega$$
$$W^+_{\lambda C} = W_{\lambda C} + \eta Error_C in_\lambda$$
$$W^+_{\Omega C} = W_{\Omega C} + \eta Error_C in_\Omega$$

Finally

We hope you've enjoyed this trip around the world of neural nets enough to try experimenting with some yourself. Listed below are some basic references, which are a good starting point for further exploration.

(020324-4)

Acknowledgements

The authors would like to express their gratitude to Ann B. Reddipogu, Niccolo Capanni and Sethuraman Muthuraman for their help putting together these articles.

Further reading

Some popular books on Neural Nets for those who'd like to delve deeper:

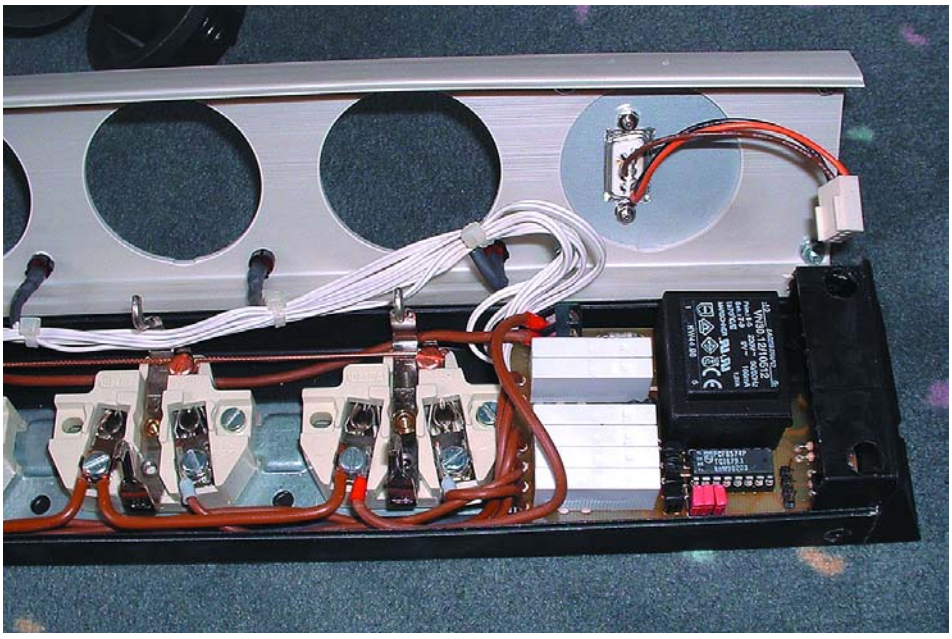
K. Gurney, **An Introduction to Neural Networks**, UCL Press, 1997.
I. Pratt, **Artificial Intelligence**, MacMillan, 1994.
P. D. Wasserman, **Neural Computing: theory and practice**, Van Nostrand Reinhold, 1989.

Switched-Outlet Power Bar with RS232 Control

Remote switching using RS232 and I²C

Design by W. Frank

This circuit can be built into a power bar and used to remotely switch six socket-outlets. It is controlled via an RS232 interface and uses an I²C-bus IC (type PCF8574P) to decode the RS232 signals.



A remotely switchable power bar controlled by a PC via the RS232 serial interface can be realized with a minimum of hardware and software drivers (for use under Windows, DOS and Linux) by using an IC to decode the RS232 signals and individually control each of the socket-outlets.

The 8-bit I/O expander

As can be seen from the block diagram in **Figure 1**, the IC in question, a type PCF8574, is basically a serial-to-parallel converter. What it essentially does is to take serial data from the I²C bus and feed them into a shift register with an 8-bit output latch. Of course, this is nothing special, and it can be done using standard components.

However, the PCF8574 has a number of useful features that make it a good choice this application. For instance, it is addressable. Three bits of the PCF8574's address are freely selectable, while the four most significant bits are either '0100' (for the PCF8574) or '0111' (for the PCF8574A version). This means that in principle, up to 16 such devices can be independently operated using a single I²C bus.

The PCF8574 also has an internal power-on reset circuit, so the output latch assumes a defined state after the circuit is switched on. The I²C bus signals pass through an input filter and thus arrive at the control logic of the IC free of interference.

Ports P0-P7 are not just outputs. They operate in a 'quasi-bidirectional' mode, so the levels on P0-P7 can not only be set in the write mode, but also read in the read mode, without having to use a con-

trol signal to select the data direction. However, this capability is not used in this circuit.

The rest of the hardware

Besides the PCF8574, there's nothing particularly remarkable about the schematic diagram of the circuit shown in **Figure 2**. The base address of the IC can be set using three jumpers (JP1-JP3). If you are sure you will never want to change the address, you can omit the pull-up resistors (R1-R3) and simply connect the IC pins directly to ground or the positive supply voltage.

The I²C bus is emulated by the RS232 interface of the PC, which is connected to K6. Besides the ground lead, only the DTR and RTS leads are used for this purpose. The diodes provide level conversion from ±12 V (RS232) to the +5 V and 0 V that the PCF8574 needs for the clock and data signals. This simple solution permits only one-way data traffic, from the PC to the PCF8574, but that's all we need here.

Six of the chip's eight I/O pins are used here, with each one controlling a relay via a driver transistor. Freewheeling diodes are connected in parallel with the coils of the relays. The supply voltage for the relays is approximately 12 V, which is taken directly from the 'raw' DC voltage obtained by rectifying the secondary voltage of power supply transformer Tr1 using B1. A voltage regulator provides +5 V for the PCF8574.
LEDs with 1.2-kΩ series resistors can optionally be connected from the junctions of the relay coils and the collectors of the driver transistors to the +12-V line, as shown on the schematic diagram, in order to indicate the status of each of the relays.

Construction and fitting

The layout of the printed circuit board for this project is shown in **Figure 3**. The dimensions of the circuit board have been chosen to allow it to be easily fitted into commonly available multi-way power bars. Naturally, it must be possible to take apart the power bar. Cheap power bars, which are usually hot-welded together, are thus not suitable. Also, the individual socket-out-

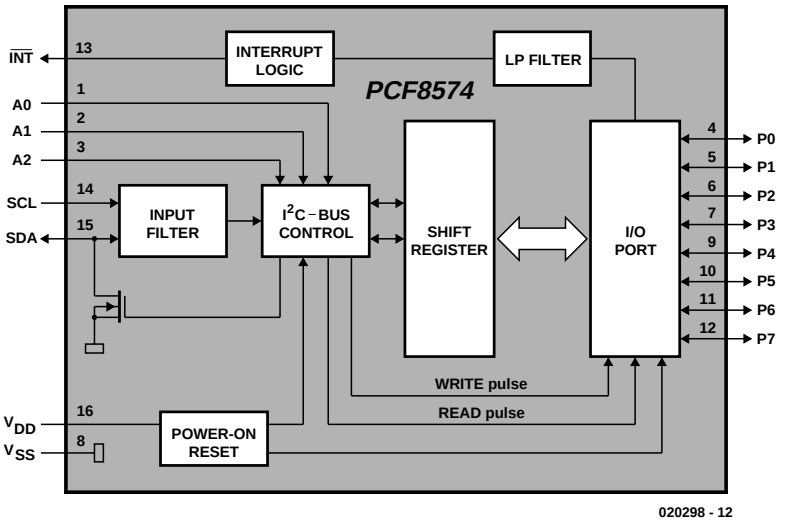


Figure 1. Functional block diagram of the PCF8574 (Philips Semiconductors).

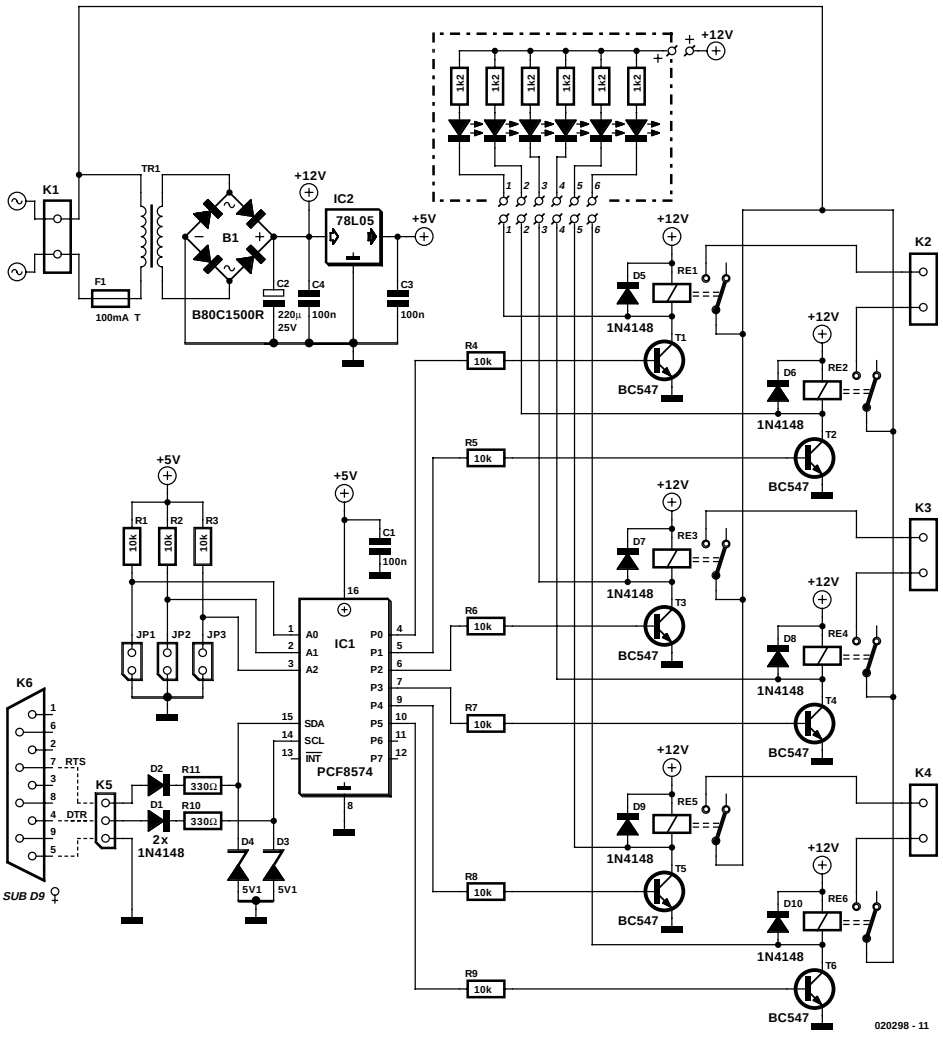


Figure 2. The circuit combines the I²C chip with a serial interface and six output relays.

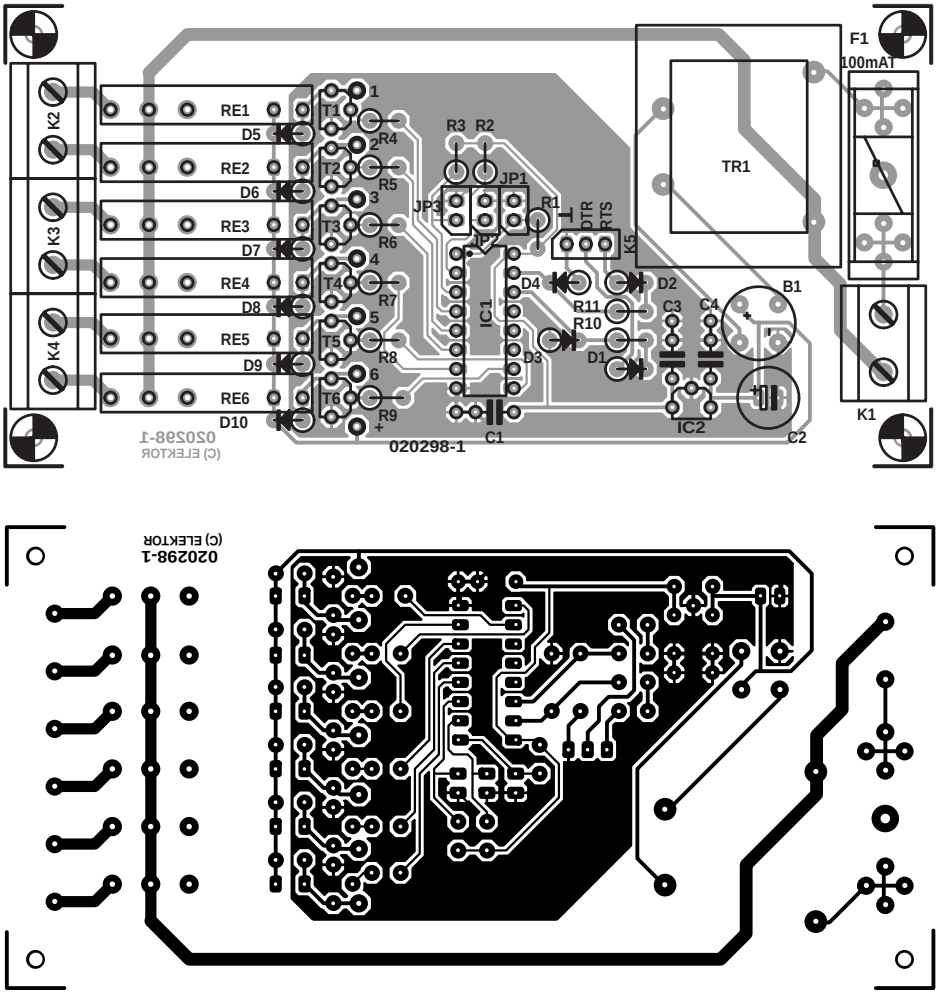


Figure 3. The printed circuit board fits exactly into a power bar enclosure.

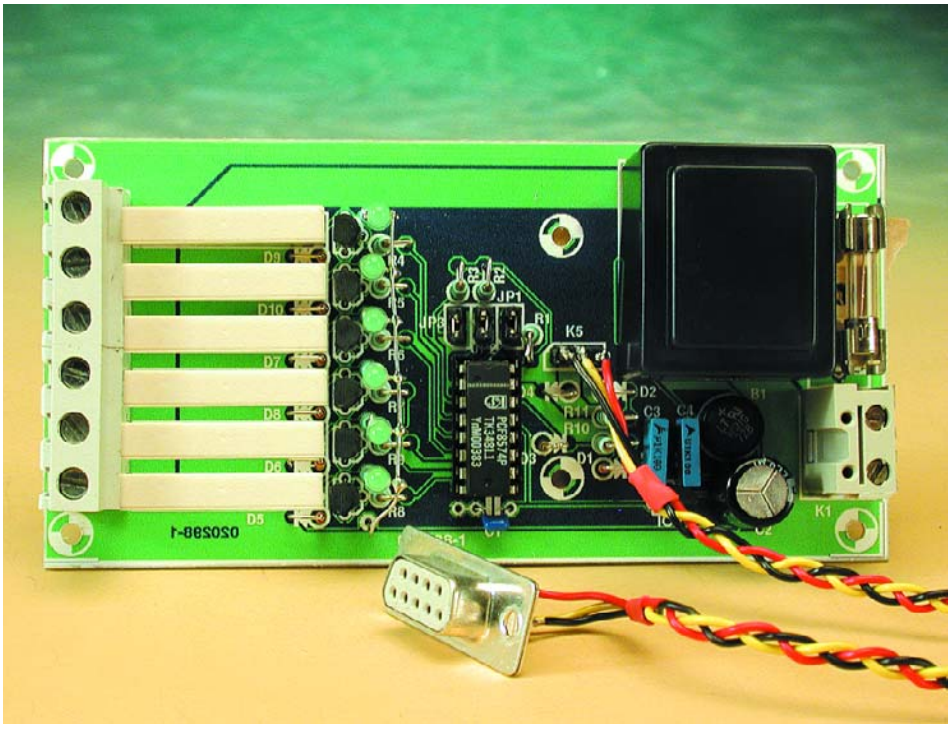


Figure 4. The fully assembled *Elektor Electronics* prototype circuit.

COMPONENTS LIST

Resistors:
R1-R9 = 10kΩ
R10,R11 = 330Ω

Capacitors:
C1,C3,C4 = 100nF
C2 = 220µF 25V radial

Semiconductors:
B1 = B80C1500 in round case (80V piv, 1.5A continuous)
D1,D2,D5-D10 = 1N4148
D3,D4 = zener diode 5.1V, 500mW
T1-T6 = BC547
IC1 = PCF8574 or PCF8574A
IC2 = 78L05

Miscellaneous:
JP1,JP2,JP3 = jumper or wire link
K1-K4 = 2-way PCB terminal block, lead pitch 7.5mm
K5 = 3-way SIL pinheader
K6 = 9-way sub-D socket (female), chassis mount
TR1 = mains transformer, 9V 1.5VA, (e.g., Hahn BV EI 302 2021)
RE1-RE6 = relay, Finder type 34.51.7.012.00 (12V) (Conrad Electronics # 504459)
F1 = fuse, 100mA, slow, with fuse holder and protective cover
Optional: six 1.2kΩ resistors and six LEDs
PCB, order code **020298-1** (see Readers Services page)
Disk, order code **020298-11,** or Free Download)

lets must be interconnected using flexible wiring, rather than solid strips of metal. To make room for the control circuit, remove the first of the socket-outlets. The circuit board has four drilled mounting holes to allow it to be securely attached to the power bar enclosure.

The photo at the head of the article shows the author's prototype. The circuit board in this photo is not the same as the *Elektor Electronics* circuit board (see **Figure 4**).

Assembling the circuit board is perfectly straightforward. All of the passive components and discrete semiconductors are fitted vertically. As already noted, the three jumpers can be replaced by wire links to ground or the 5-V supply voltage. In

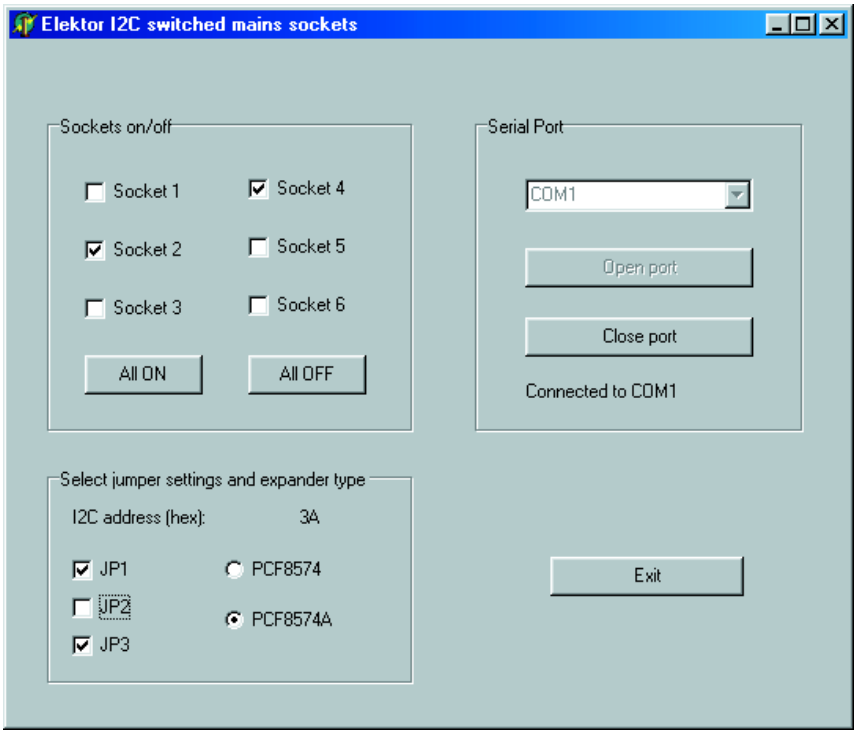


Figure 5. Using a Delphi 5 window to control the power bar under Windows.

this case, R1-R3 can be omitted. The narrow mains relays made by the Swiss company Finder allow the board to be kept very compact.

Drivers

The project includes source code and .exe files for Windows (a Delphi application), Linux and DOS (both written in C). All of these programs can be downloaded free of charge from the *Elektor Electronics* website, including the source code. They can also be obtained from Readers Services on diskette (order number **020298-11**).

The core of the program consists of an emulation of the I²C bus using two leads of the 'normal' serial PC interface. The DTR lead drives the serial I²C clock line SCL, while the RTS lead drives the I²C data line SDA. Since the PC cannot receive any data from the I²C chip, the software simply ignores Acknowledge signals from the PCF8574.

After the PC software is started, or after a data packet has been sent, the bus is in the idle state, with both of the I²C lines high. The bus master initiates a data transfer by generating a negative edge on SDA while SCL is high (Start condition). After this, SCL also goes low. The seven address bits, followed by the R/W bit, are transferred sequentially by a series of rising edges on the SCL line. The level on the SDA line is only allowed to change while the SCL line is low. In this application, the R/W bit is naturally always 0 (= write).

Normally, the slave is allowed to confirm the receipt of a valid address by sending Acknowledge. Although this is not the case here, the master must still generate an extra clock pulse, which the slave interprets as confirmation of the Acknowledge.

After this, the slave is ready to receive the eight data bits. Each bit is directly associated with one of the IC outputs. The data bits are followed by an extra SCL pulse to confirm the Acknowledge the slave thinks it has sent, after which the master terminates the transfer with a positive edge on SDA while SCL is high.

Control

The functions of `SERIALDLL.DLL`, which has been extensively described in the March 2003 issue of *Elektor Electronics*, are used to drive the serial interface. This DLL, which also forms part of the software for this project that is available by download or on diskette, should be copied to the directory \WINDOWS\SYSTEM.

The actual program, `SOCKETS`, uses the functions SetDTR, Reset-DTR, SetRTS and ResetRTS. In addition, COMPortExist determines which serial ports are available, and OpenCOM & CloseCOM are necessary to active and deactivate the port to which the power bar is connected.

The most important procedure is SendI2C, which sends a data byte to a particular I²C address. A glance at the source codes shows that all that this involves is driving the two lines in the proper sequence. A brief delay is introduced after each level change on the lines in order to keep the data transmission rate on the bus within allowable limits.

Figure 5 shows the window displayed by the sample Delphi 5 program, which is available as source code. It is fairly self-explanatory. At the beginning, the desired COM port is selected and opened, after which the address of the I²C chip is selected. This depends on the IC type (the PCF8574 has a base address of $20, while the -A version has a base of address of $38) and the configuration of jumpers JP1-JP3. Once these settings have been made, the individual socket-outlets can be switched on or off as desired. To stop the program, close the serial port or simply click on Exit.

(020298-1)

Free Downloads

– source code files and .exe files for Windows (Delphi application), Linux and DOS (both written in C).
File number: **020298-11.zip**
– circuit board layout in PDF format.
File number: **020298-1.zip**

www.elektor-electronics.co.uk/dl/dl.htm
(select month of publication)

New C Compilers: SDCC and AVR-GCC

Freeware for MCS51 and AVR!

By G. May, DL3ABQ

Alongside a wide variety of expensive, up-market products, two reasonably powerful C compilers for microcontrollers have emerged in the freeware circuit.

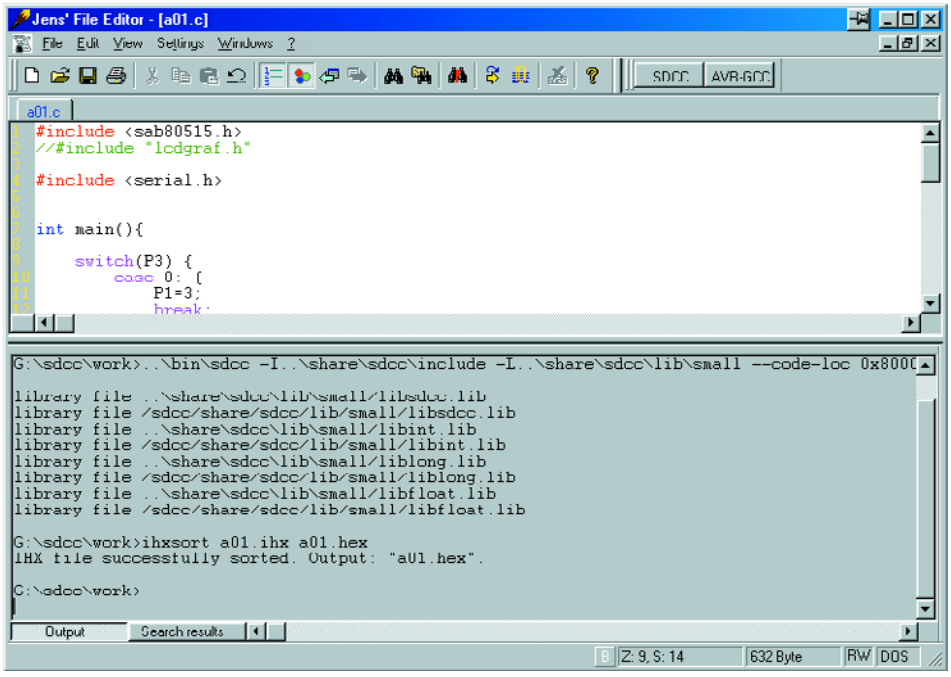


Figure 1. JFE in use.

The use of 'free' C compilers often presents unexpected problems because of the frequent lack of a user-friendly user interface. Also, despite their unrivalled price/performance ratio, these programs deter many potential users because their configuration and setup is not without problems. This article aims at providing some information to help overcome such initial problems. Specifically, we hope to supply some practical guidance in relation to the 'SDCC' compiler for the MCS-51 series and 'AVR-GCC' for Atmel micros.

Editors

Because both compilers are in essence command-line driven programs, you will need an editor (word processing) utility to start with. This editor should allow programs to be written in a comfortable way. There are some excellent, free, editors like JFE (Jens File Editor) and Proton, which offer roughly the same functionality.

Both editors support syntax highlighting (colouring of recognised 'C' terms) which is a great asset for spotting typos in your programs. They also allow external command line tools (like the C compilers discussed here) to be embedded into the user interface.

Such an editor enables you to write your own microcontroller program and compile it by calling the command line compiler. Once the compiler has finished its job, the resulting object code is ready for transferring (downloading) to the target microcontroller. The process of embedding the compilers into the editor is really simple and well described by the Online Help function.

Figure 1 shows JFE being used. As you can see, both SDCC and AVR-GCC have been embedded (buttons to the right of the menu). A program has just been compiled by SDCC and its output is shown by JFE in the lower area of the window.

Figure 2 shows the settings required for calling SDCC from JFE.

SDCC

SDCC (Small Device C Compiler) allows you to compile C programs for

devices from the 8051 series and a couple of other processors. The program is launched from the command line level by typing

```
sdcc <options> <file name>
```

Provided no errors are found, SDCC generates a file with the extension 'ihx', which contains the machine (object) code in IntelHex format.

Unfortunately, some monitor programs may run into problems because the lines in the IntelHex file have not been incrementally sorted by memory address. In order to overcome this difficulty, the author has written a small utility called `ihxsort` that performs this sorting operation afterwards. The program may be obtained from the Free downloads section of the *Elektor Electronics* website under number **020339-11.zip**, which contains the source code as well as the executable (Windows) program. To prevent tedious typing over and over again it is recommended to write a batch file that calls the compiler with the relevant command line options. Of course, the parameters depend on the microcontroller system used (monitor program, memory configuration, etc.) as well as on the installation paths assigned to SDCC. Here is a example of such a batch file:

```
(compile.bat):

c:\sdcc\bin\sdcc
-Ic:\sdcc\share\sdcc\include
-c:\sdcc\share\sdcc\lib\small
—code-loc 0x8000
—stack-loc 0x20 %1.c
ihxsort %1.ihx %1.hex
```

Calling the compiler requires this command line

```
compile <C source file without extension>
```

SDCC expects this batch file in the path `c:\sdcc`, compiles the program according to the `small` model for code position 8000h (e.g., for a monitor program, with stand-alone applications the start address is usually 0000h) and then arranges for the stack pointer to point to address 20h. The resulting `ihx` file has the same name as the source file but the extension `.ihx`. At the same time, the program `ihxsort` is called and launched for the sorting work, producing an output file with the extension `.hex`.

The options available with the command lines for SDCC are numerous and a full overview is unfortunately beyond the scope of this article. They may, however, be found in the user manual file `sdccman.pdf` contained in the previously mentioned zip file.

The speed of the generated code is not up to that of commercially available compilers like the industry standard Keil product. According to the manual, SDCC does some optimising during the compilation phase, but this is difficult to track or even prove in practice. None the less, SDCC is a good performer although it must be said that there is no substitute for assembly code in the case of time critical routines.

There is considerable variation in the quality of the library files supplied with SDCC, so the files should only be used after careful reading of the user manual mentioned above. Some example programs are supplied that allows the user to do a couple of quick tests to get started. Finally, there's a mailing list which you may join to discuss any problems that may occur.

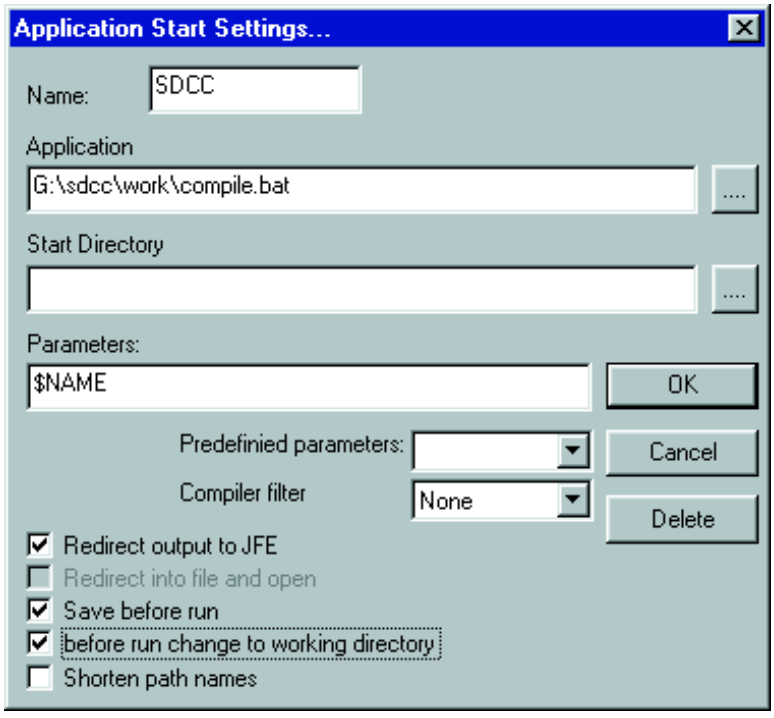


Figure 2.Settings to embed SDCC into JFE.

AVR-GCC

AVR-GCC is the result of 'migrating' GCC (well known among UNIX fans) to Atmel AVR RISC controllers. The new compiler has seen rapid spreading across the Internet. Its machine code output is very powerful and comparable in performance with that produced by commercially available compilers.

With the Windows version, it is necessary to launch `run.bat` in the AVR-GCC subdirectory in order to set up a couple of important parameters for AVR-GCC. This happens automatically when the program is launched via the Start menu entry. Next, the user is free to go to the subdirectory containing his/her program and have it compiled.

AVR-GCC employs Make files for project management. The settings for compiling a particular project are stored in a file named `makefile`, while the compile process proper (in the relevant directory) is launched by calling

```
make
```

There are a number of standard 'Make' files available which only require small corrections to be able to use them for your own projects. During the installation, one such file is already stored in the subdirectory `avrgcc\avrfreaks`. For the purpose of your own projects, create a new directory, copy the `makefile` 'template' into it and adapt the paths, file names and any other options to suit your requirements.

During the compile process, AVR-GCC creates a `rom` and an `eep` file. The `rom` file con-

Internet addresses

SDCC download and mailing list
http://sourceforge.net/projects/sdcc

AVR-GCC download, examples and libraries
www.mikrocontroller.net/index.en.htm

JFE download
http://home.t-online.de/home/Jens.Altmann/jfe.htm

Proton editor
www.meybohm.de

tains the program for the microcontroller's Flash memory, and the eep file, the EEPROM contents — both are written in IntelHex format.

A number of excellent libraries are available for AVR-GCC and may be picked up from the Internet, just like example programs.

To close off this article, we'd like to mention a problem many AVR-GCC have reported in the ACR-GCC forum.

Just as many PC programs start by displaying a "Hello World" message, microcontroller applications often indicate their initial activity by flashing an LED on a port pin. In many example ('demo') programs written for other microcontroller compilers, the time delay between the flashes is typically created by an empty for/next loop. Such constructs are, however, 'optimised away' by AVR-GCC. Consequently, time delays should be implemented using the controller's own timers. Alternatively, using `volatile` variables, the compiler should be prevented from removing such a loop.

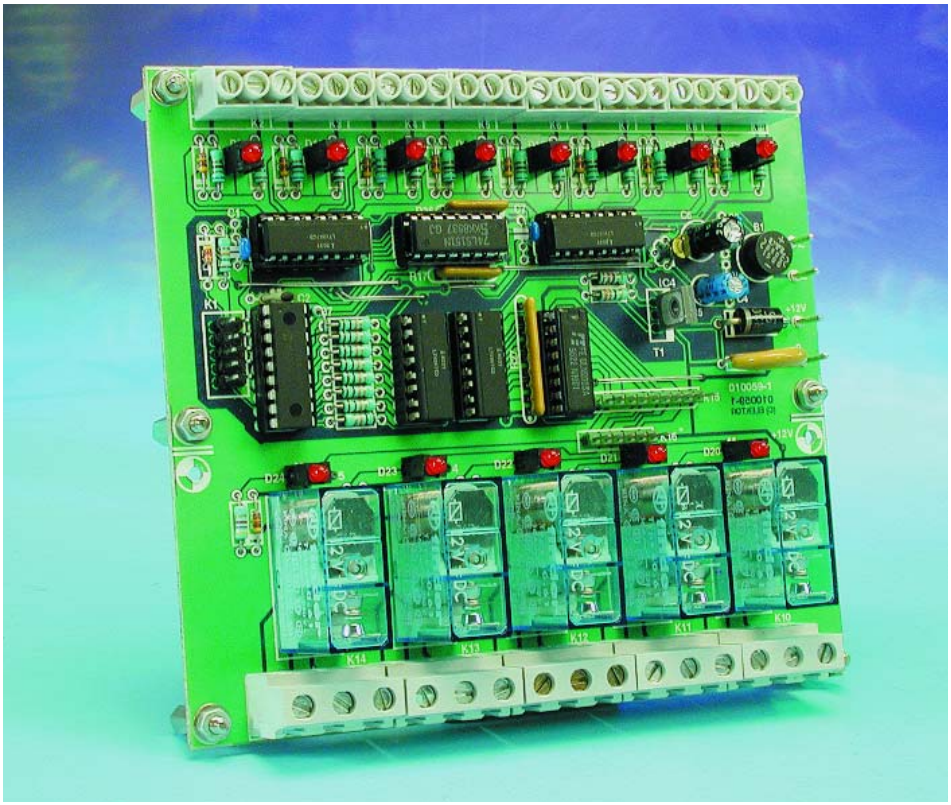
(020339-1)

Pico PLC

microcontroller or Programmable Logic Controller?

Design by H Gebhard, DF2DS

There are certainly strong similarities between these two approaches to controlling equipment, and with the circuit described here we blur any remaining distinction still further. The Pico PLC employs only standard components and the necessary software development tools are available free on the Internet for private users.



A distinction is frequently drawn between microcontrollers and PLCs (programmable logic controllers) used in industrial control applications. The first thing that is apparent is that the PLC is designed for industrial use: in particular this usually means a supply voltage of 24 V. Also, the circuit will be robust against supply polarity reversal, voltage spikes and short circuits.

We find a further difference in the software. A PLC runs a monitor program which controls the sequence of events executed. This involves, among other things, the use of interrupts, software timers and dealing with various interface protocols (frequently RS-232 or CAN bus).

The user program always executes cyclically over a fixed period of time. This ensures that the timing behaviour of the controller is predictable. An important prerequisite here is the ban on 'busy wait' loops: only in the main program loop can the user test whether a particular event has occurred.

Various methods are suggested for programming PLCs, such as the 'ladder diagram' (which looks like a set of interconnected switches) or as an 'instruction list' (which looks more like assembler source code). The user program is stored in the PLC in flash memory and is run automatically when power is applied.

Is a microcontroller better than a PLC?

There is no completely satisfactory answer to this question. Microcontroller-based systems are consider-

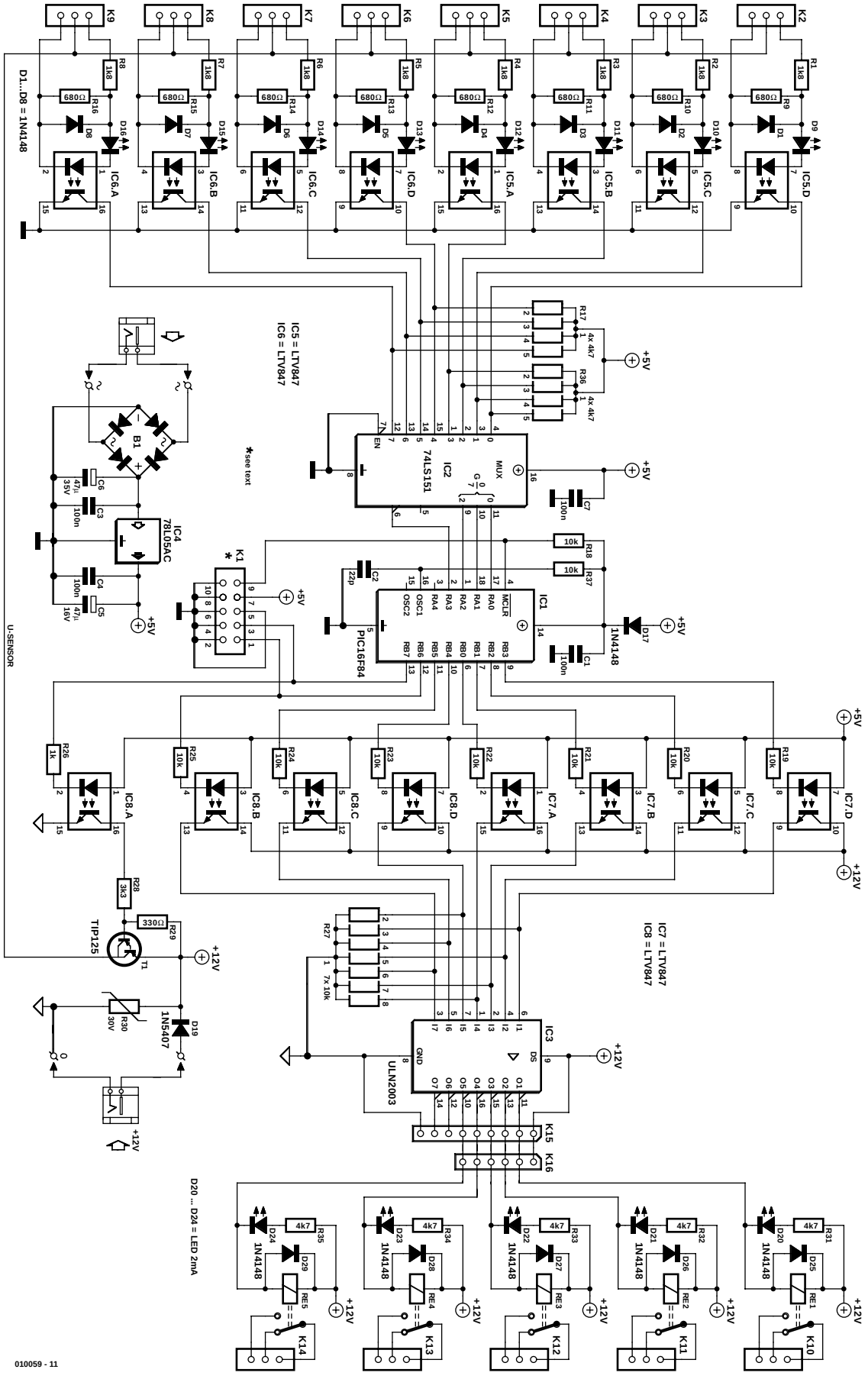


Figure 1. A microcontroller with isolated inputs and outputs.

ably more flexible, but require much more programming experience and discipline when used as PLCs. The possibilities are much wider, since the full range of assembly-code instructions is available for use. Almost always a high-level language compiler is available, generally C or BASIC, although sometimes Pascal or other more exotic languages are found.

It should however be clear to any developer that in using a compiler for a high-level language it will not be possible to squeeze the last clock cycle of performance out of the microcontroller. However, programming in a high-level language does confer the advantages of simplifying debugging and modification or adaptation of the code, when compared to using assembler.

The Pico PLC

The Pico PLC system can be used in both application areas. At the heart of the hardware is a Microchip PIC16F84 microcontroller [1]. This microcontroller offers 13 digital I/O port lines, 64 bytes of EEPROM and 1024 14-bit words of flash program memory. At first sight this seems laughably small compared to the amount of memory found in a PC, but thanks to the specialised highly-efficient instruction set it is nonetheless adequate for many applications.

If the microcontroller's tasks are not time-critical, the quartz crystal can be dispensed with and the processor clock generated using its internal RC oscillator (with R37 and C2).

As can be seen from the circuit diagram in **Figure 1**, there are no secrets lurking inside the processor. The inputs are brought one-by-one to port pin RA3, which is configured as an input, via multiplexer IC2. The selection of the input is done using port pins RA0, RA1 and RA2. The inversion at the output of the multiplexer cancels out the inversion which occurs in the optocouplers.

The individual inputs are electrically isolated using optocouplers, as demanded by standard PLC hardware design practice. The potential dividers prevent too great a current flowing through the optocoupler with a high input voltage (here between 10 V and 30 V). The diode connected in antiparallel with the optocoupler is important: it protects the LED in the coupler from a reverse polarity input. The infrared LEDs used have a low maximum reverse voltage (6 V maximum for the PC847 [2] or LTV847).

A special feature is the *U sensor* connection, which is an output driven by the microcontroller. Using this the sensors can be supplied with power only when required. This power-saving feature is particularly useful in battery-operated systems.

The output drivers are also electrically isolated from the controller

COMPONENTS LIST

Resistors:
R1-R8 = 1kΩ8
R9-R16 = 680Ω
R17,R36 = 4-way 4kΩ7 SIL array
R18-R25 = 10kΩ
R26 = 1kΩ
R27 = 7-way (or 8-way) 10kΩ SIL array
R28 = 3kΩ3
R29 = 330Ω
R30 = varistor 30V, 600mW, diameter 15-17 mm (e.g., BC-Components # 2322 5953006)
R31-R35 = 4kΩ7
R37 = 10kΩ (4kΩ7)*

Capacitors:
C1,C3,C4,C7 = 100nF ceramic
C2 = 22pF
C5 = 47µF 16V radial
C6 = 47µF 35V radial

Semiconductors:
B1 = B80C1500 (80V piv, 1.5A), in round case
D1-D8,D17,D25-D29 = 1N4148

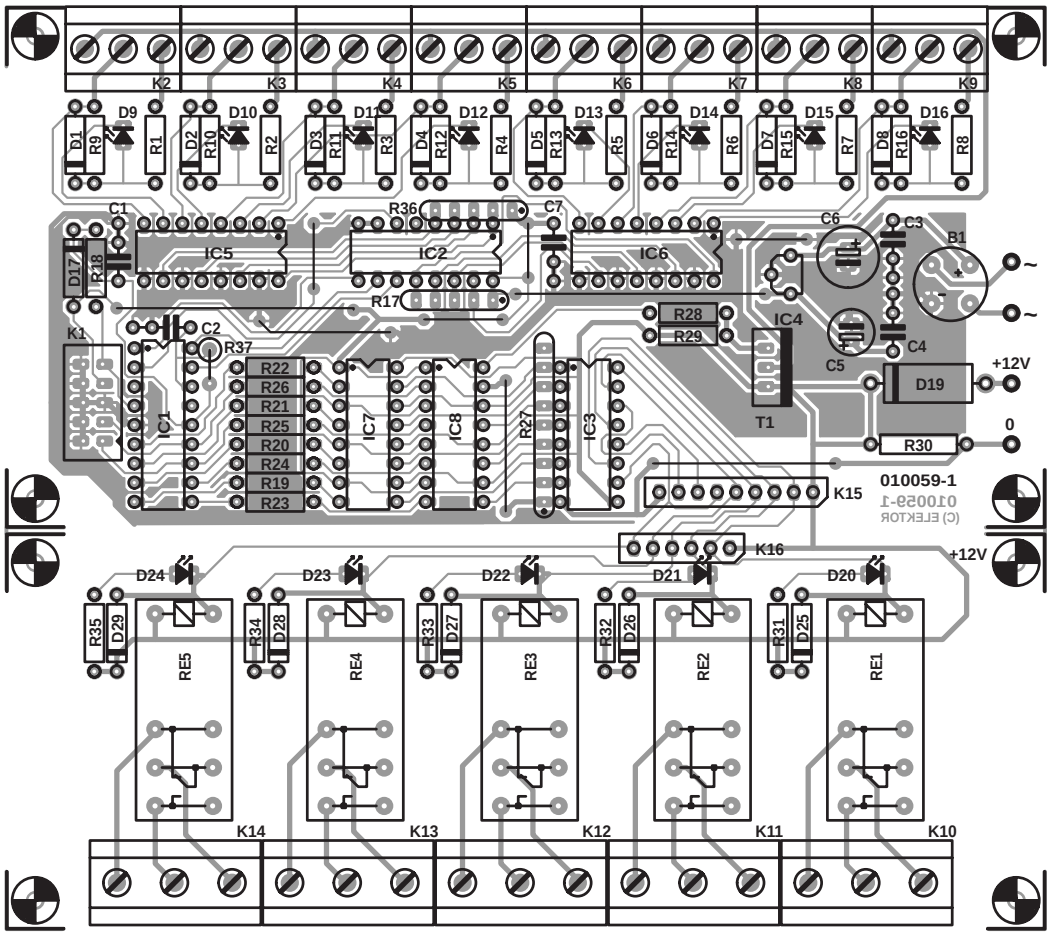


Figure 2. The part of the circuit board carrying the relays can be separated from the rest of the PLC.

using optocouplers. The desired output voltage can simply be connected up: it will generally be either 12 V or 24 V. IC3, a ULN2003, allows output voltages of up to 50 V, as long as the total output current does not exceed 500 mA.

Since it will be desired to switch greater loads than this, as well as mains voltages, relays are connected to the outputs of the power driver IC, each capable of switching up to 16 A at 250 V AC. This should be sufficient for almost all applications; and it is always possible to omit the relays if the drive of the ULN2003 is enough by itself.

The software

Any item of microprocessor-controlled hardware can only be as good as the software running on it, and the quality of that software depends in turn on the development tools used to write it. One of the main reasons for selecting the PIC16F84 was the availability of good-quality, low-cost (or even better, free) development tools. The manufacturer, in line with most of the competitors in the market, offers a suite of software development tools. MPLAB [4] from Microchip includes an assembler, a linker and a simulator, which allows the logical correctness of the software to be verified on a PC. Together, these tools allow a huge range of possible projects to be developed, as a glance at the Microchip website will clearly show.

If you would prefer to program the PIC in a high-level language, a suitable compiler will be required. Fortunately two good C compilers for the PIC are already available, namely C2C by Pavel Baranov [5] and CC5X from Knudsen Data [6]. The compilers differ slightly as regards processor-specific instructions, but otherwise are broadly as good as one another. Assembler source code is generated from a C source file; the result is automatically assembled and converted into a binary file (in Intel Hex format). Now we have just one problem.

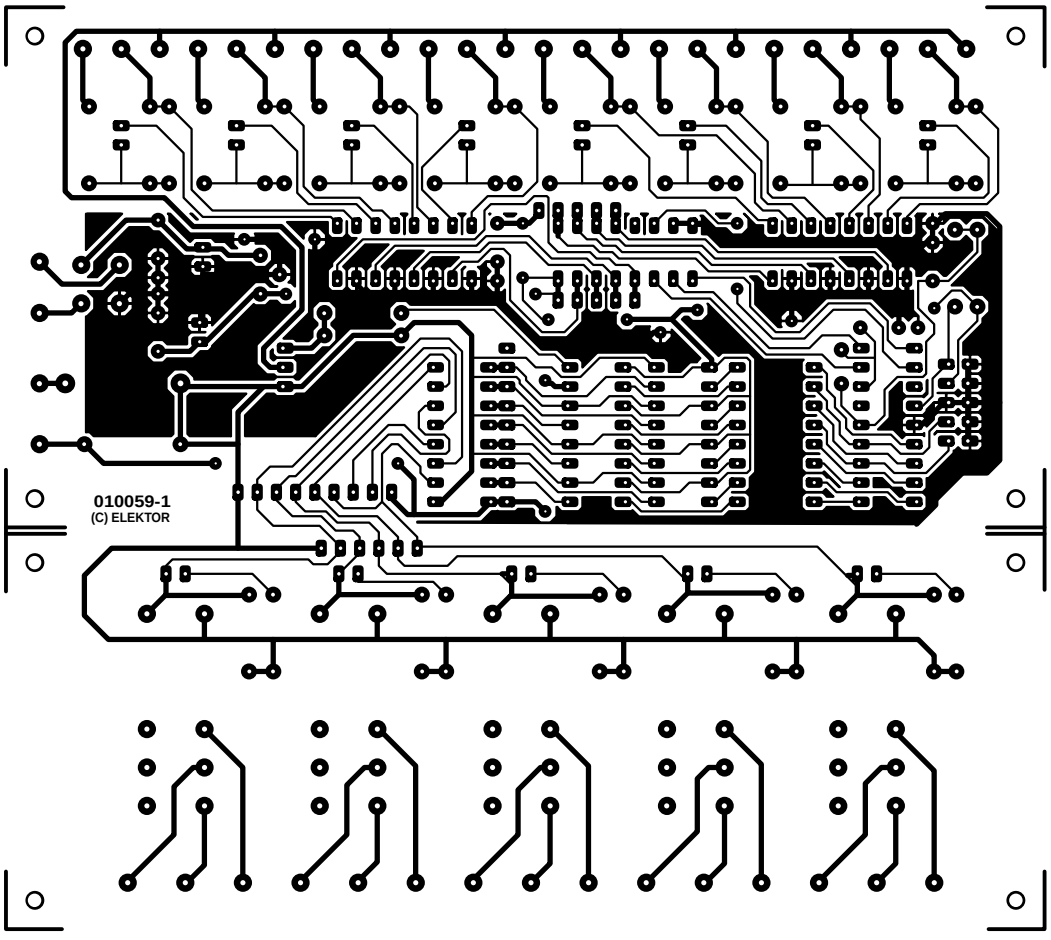
D9-D16,D20-D24 = LED. low current
D19 = 1N5407
IC1 = PIC16F84A-04/P
IC2 = 74LS151
IC3 = ULN2003
IC4 = 78L05AC
IC5-IC8 = LTV847 (Liteon), ILQ621 (Infineon) or PC847 (Sharp)

Miscellaneous:
K1 = 10-way boxheader
K2-K9 = 3-way PCB terminal block, lead pitch 5mm
K10-K14 = 3-way PCB terminal block, lead pitch 7.5mm
K15 = 9-way pinheader
K16 = 6-way pinheader
RE1-RE5 = relay, 16 A/250 VAC (Finder # 40.61, coil 12VDC, 220Ω; or Omron # G2R-1-E 12VDC; or Schrack # RP310012)
PCB, order code **010059-1** (see Readers Services page)
Disk, test program, order code 010059-11 or Free Download from www.elektor-electronics.co.uk

How do we get the program into the PIC?

The PIC microcontrollers are popular not least because of their ease of programming. Since PICs can be programmed serially the interface is relatively straightforward and in principle a simple cable is all that is required to do the job. But even a more luxurious programmer does not involve much hardware: for example, a classical 'low cost' development system was presented in the July/August 1998 edition of *Elektor Electronics*. The tiny evaluation board described there provided a stabilised power supply and a switchable and variable clock oscillator, as well as a small prototyping area, where for example LEDs could be fitted to observe the levels on the output ports.

An alternative programmer circuit board can be found fully documented in [3]. The necessary (DOS) program can also be obtained from the website.



Construction and installation

The construction of the Pico PLC on the circuit board shown in **Figure 2** should present no problems. Components are only fitted on one side, and all components are of the normal through-hole type rather than surface mount devices. If the relays are not required the circuit board can be cut between K15 and K16 and the PLC can then be built into a smaller enclosure. It is also possible to fit the part of the circuit board carrying the relays elsewhere in the enclosure, connecting it to the PLC via K15 and K16. If the relays are fitted, the safety precautions for class 2 devices must be observed at the outputs.

The ICs should of course be fitted in sockets and good-quality components should be used. To avoid loose connections, use tension clamp mounting terminals on the inputs and outputs.

No special software is provided for the Pico PLC though Readers' Services other than a short test program. The program was created using the free CC5X tools and MPLAB, and creates a kind of running light display from D20 to D24 and back — assuming the hardware is working correctly. Then D24 starts to blink, and the state of the inputs is displayed. While D24 is extinguished D20 to D23 show the states of the four lower-numbered input bits, while when D24 is lit, they show the states of the four higher-numbered inputs.

There are many applications for the Pico PLC. The original use was to control the drive mechanism for an amateur radio antenna mast. Other possibilities include controlling the shutters or Venetian blinds for a window, heating control or automating mechanical toys and models.

(010059)

References and links

[1] Microchip: www. microchip. com

[2] Datasheet available (for example) at www.sharpmeg.com/products/opto/pdf/pc847x.pdf

[3] Madsen programming adapter www.jdm.homepage.dk/newpic.htm

[4] MPLAB: development environment for Microchip microcontrollers www.microchip.com/1000/pline/tools/picmicro/devenv/mplabi/index.htm

[5] Baranov, Pavel: C2C-Compiler www.geocities.com/SiliconValley/Network/3656/c2c/c.html

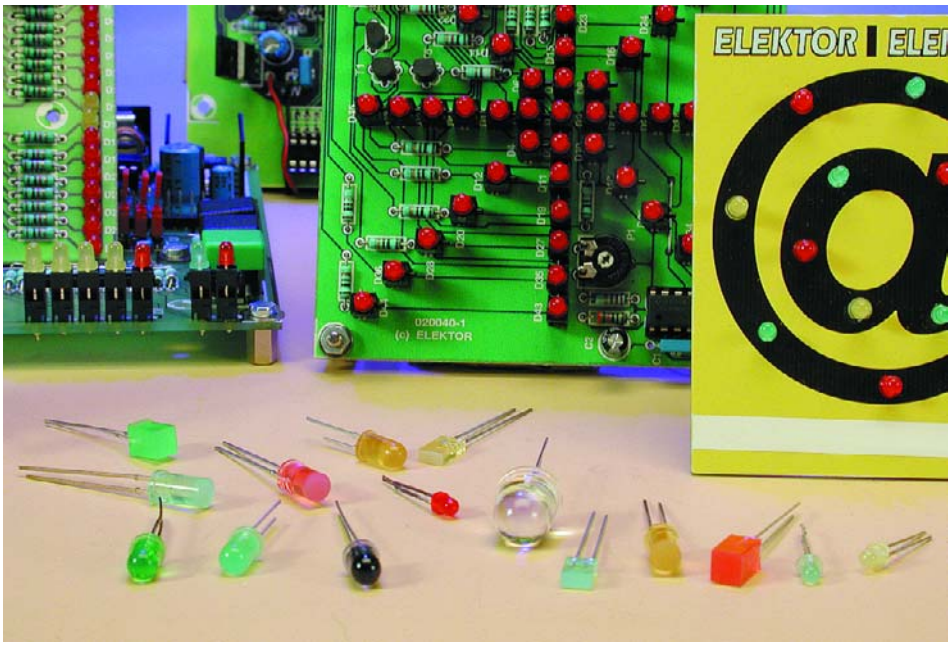
[6] B Knudsen Data: CC5X www.bknd.com/cc5x/index.shtml

LED Arrays

matrixed clusters for lighting purposes

By K.-J. Thiesler

A handful of high-intensity LEDs is sure to provide new accents in room lighting. In traffic lights and other signalling equipment, they save energy and reduce maintenance cost thanks to their long life expectancy as compared with conventional lamps.



Manufacturers have been able to push the light intensity of modern LEDs to levels where these devices can be used for lighting (illumination) purposes, even if prices are still relatively high. However, LEDs do offer some clear advantages over traditional electrical light sources. For one thing, their life expectancy of up to 100,000 hours is impressive. Also, the optics are integrated in the enclosure to ensure the emitted light is bundled to requirements. A further advantage is the fast on/off time of a LED. In cars, for example, a LED brake light gives a clearly faster response than a conventional bulb. Whereas LEDs respond within 100 ns or so, 'cold' bulbs need to be pre-heated for about 100-300 ms. During that period, they require current peaks of up to 50 times their nominal value.

Of course, there are also disadvantages to the use of LEDs. Bulbs emit lots of infrared light which allows them to keep functioning at high temperatures. LEDs, on the other hand, are passive semiconductors with an associated temperature spectrum. Their high power density causes the maximum chip temperature of 125 °C to be reached at an ambient temperature of about 85 °C. Such a temperature is easily reached, for example, inside a third break light on a bright sunny afternoon in summer. When the chip temperature exceeds 125 °C, the LED stops working as soon as it lights up. This usually happens at the maximum permissible forward current which causes the chip temperature to rise even further. Because the power handling capacity of a semiconductor is inversely related to ambient temperature, worst-case conditions easily arise in simple circuits without compensation measures. The result: the overloaded LED no longer lights and goes open-circuit.

In the manufacturing process of LEDs, the deviations in electrical performance is such that devices may be gathered in batches with a lower tolerance. For a single indicator LED used in a standard circuit configuration, some deviation is perfectly acceptable. Not so, however, in the case of LEDs being clustered into arrays and positioned closely together. In arrays, individual LEDs are also connected-up to form sub-clusters, which causes them to elec-

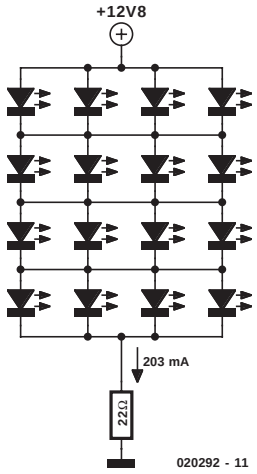


Figure 1. LED array using just one series resistor.

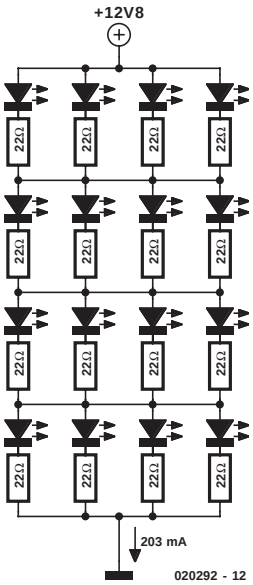


Figure 2. Here, each LED has its own current limiting resistor.

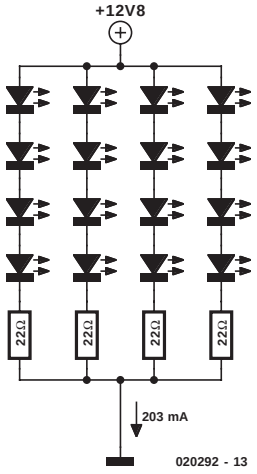


Figure 3. Four independent strings of LEDs, each with one series resistor.

trically 'compete' with each other. A tolerance of 150 mV in the forward voltage specification is not unusual. A finer division and classification is simply not economical. If optical performance tolerance is the yardstick, for example, in the design of multdigit LED displays, LEDs need to be selected for luminosity by eye or by means of optical instruments. For pure lighting purposes, such a time-consuming selection process is not necessary.

Current tolerances

In general, three basic design variants are available when you want to cluster a handful of LEDs into an array which is to be used for lighting purposes.

Figure 1 shows the variant with the array powered through a single series resistor. A very workable solution which also functions with LEDs 'from the margin', i.e., having a forward drop tolerance of up to 150 mV between individual LEDs in the matrix. After all, a small rise in forward current is coupled to a proportional rise in forward voltage. In this matrix, some LEDs having the upper tolerance value will light less brightly than others with a lower forward drop. If the circuit is designed for 50 mA per LED, the forward current of individual LEDs will be between 40 mA for LEDs with the highest, and 62 mA for LEDs with the lowest forward drop.

The advantages and disadvantages of the circuit shown in Figure 1 may be summarized as follows:

When one LED fails, the others will continue to work, albeit with a shorter life expectancy.

The design s simple — only one series resistor is required. The LEDs being connected up in series and parallel, the PCB design remains uncluttered.

Failure detection will only work in case the entire array fails. In the automotive industry, legislation dictates failure indication for direction indicators, which can not be realised using this type of array because the series resistor can not be employed as a sensor.

In **Figure 2**, the LEDs are configured in a different kind of matrix marked by individual series resistors for each LED. Obviously, the resultant PCB will be large and densely populated. The series resistors with each LED makes it less dependent on tolerances of other LEDs. Also, the current 'band' is narrowed down: 46 mA (min.) to 53 mA (max.). The series resistors allow the current through each LED to be dimensioned with sufficient accuracy. This variant is the best for room lighting because failure of LEDs may be observed and their replacement evaluated for cost and effort. Advantages and disadvantages of the circuit:

1. When one LED fails, the others continue to function without increased death risk. The total luminosity achieved by the array will hardly suffer.

2. Because failure detection of individual LEDs is far to complex, such a function may be implemented using current/voltage detection on the power regulating device.

In **Figure 3** we've sketched a series configuration with a series resistor for each LED. There are no junctions that force a defined voltage upon each LED row. The four LED columns operate independently. Because the differences in forward voltage remain relatively small at the same forward current (as compared with forward current differences at the same forward voltage), measurement results obtained from this constellation will resemble those of the matrix circuit with resistors for each individual LED. However, the final circuit and PCB layout will be much simpler than with the circuit in **Figure 2**. The series connection into multiple strings does have a disadvantage which will be discussed further on. In a prototype of this circuit, a minimum and maximum current was measured of 47 mA and 53 mA respectively.

The string circuit does not have 'current diversion' abilities because there are no junctions of parallel connected LEDs. If one LED fails, so does the entire string — an obvious disadvantage for room lighting because the resultant light intensity will drop considerably. Also, it makes no sense to replace the entire array just because one LED has failed.

In case of the automotive industry, suitability aspects look rather different. Failure of an LED column is not worrying because the warning indicator functionality is still warranted even at reduced light intensity. Also, the life expectancy of the LEDs in the other columns is not affected by failure of one string. A simple failure detection may be

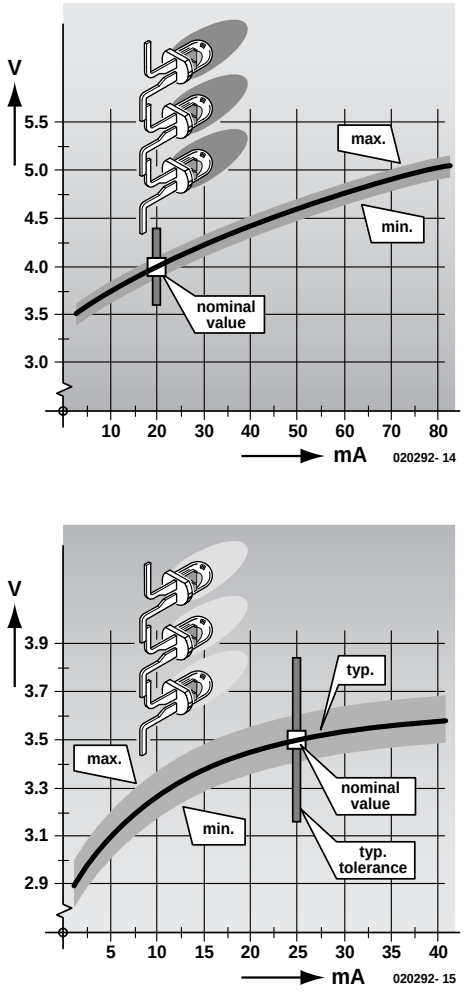


Figure 4. Design data of blue and white LEDs.

realised by employing the series resistors as current sensors.

When one LED fails, so does the entire string. The other strings, however, continue to work normally and the life expectancy of the LEDs in them is not affected.

When the array is used as a direction indicator, brake light or reversing light, failure detection is relatively straightforward since the few series resistors may be used as current sensors.

Conclusion: for all designs, it is important to realise what happens when one LED fails. Current overloading coincident with excess chip temperature causes a LED to be electrically damaged to the extent that an open-circuit occurs. In the matrix circuit shown in Figures 1 and 2, this has dire consequences because the 'adjacent' LEDs are forced to operate at worse conditions. In the row that contains the failed LED, the other paralleled LEDs are strained to handle the current of the failed LED. The fewer rows and columns that make up the array, the graver the effects of the increased current. This is acceptable when the LEDs are operated way below their maximum current specification which, in turn, reduces the changes of failure. Alternatively, the number of LED columns is such that the 'deviation' current hardly presents an extra load to individual LEDs, as compared with the consequences of the 150-mV tolerances.

Table 1

Blue LED TLHB4401	**Maximum values**			
Reverse voltage		UR	5	V
Forward current (DC)	at U_F = 4.0 V	IF	20	mA
Forward current, 10% on, t_p < 10 μs	at U_F = 5.2 V, T_A < 60 °C	IFSM	100	mA
Power dissipation		PV	100	mW
Operating temperature		TA	–40 to +100	°C
Thermal resistance		RthJA	400	K/W
Blue LED TLHB4401	**Nominal values**			
Luminosity		IV	32	mcd
Beamwidth	at I_V = 100 %	φ	±10	Degrees
	at I_V = 50 %	φ	±30	Degrees
Wavelength	at maximum I_V	λ	430	nm

Boosting brightness

Table 1 shows some typical design data for a blue LED. By implication, it shows the LED will light at 100% at a continuous current of 20 mA. The LED intensity may be raised when the current is increased in a certain way. This is done, for example, in remote controls in order to increase their range. However, simply increasing the continuous direct current through the device will cause its (fairly imminent) death. To achieve higher luminosity, the LED should be operated in pulse mode.

By pulsing an LED at 100 mA for 10 μs and a repeat rate of 1 kHz, the radiated intensity may be increased by a factor of 10 whilst the maximum case temperature of 65 p is used (which also boosts the directivity). If a constant current of 20 mA is set up, the LED may be on all he time. If, on the other hand, a much higher current is sent through the LED, the 'on' time should be reduced considerably. This allows the LED to cool down during the (relatively long) 'off' time. Some DC step-up regulator ICs have a shutdown input to which a PWM signal may be applied to create a brightness control. The important aspects to keep in mind are the maximum operating temperature of the LED and its maximum power dissipation. Although the datasheet indicates a value of +100 °C as a purely physical value, a more realistic value of +60 °C should be observed if the device is to be operated in the SOA (safe operating area). At +100 °C, the LED already lights with considerably reduced intensity.

Tolerances

The typical forward voltage of a white LED is of the order of 3.5 V ±10%. This is a nominal value often found in datasheets. The variation range is shown in **Figure 4**. At 20 mA forward current, the voltage drop across the LED may lie between 3.15 and 3.85 V. If this white LED is operated in current sensor mode using an uncalibrated but regulated voltage, the forward current will vary across an even larger range. This variation of the forward voltage causes considerable variation of brightness variance between individual LEDs.

To operate single or multiple white LEDs, regulator ICs like the LM2791/2, MAX1698, MAX1848, MAX1912, LT1618, LT1932, LTC3200, LTC3400 and LM2585T-ADJ are available. The use of the latter IC was demonstrated in the article

'White LED Lamp' in the July/August 2000 issue of *Elektor Electronics*. Its function in this the circuit is that of a boost regulator in control of a 10-element LED array. A particularly interesting IC is the LT1618 from Linear Technology because it offers simultaneous current and voltage feedback, plus features a disable input for LED brightness control. Besides, the LT1618 is housed in a space-saving MSOP-10 case and capable of switching at 1.5 MHz, which considerably reduces the component volume as compared with the venerable LM2585T-ADJ.

Many of these regulators may be used in current-sense mode, which has a positive effect on uniform luminosity. After all, the control of the bias current is more accurate and takes into account the specific forward voltage of each LED, whether 3.15 V or 3.85 V.

However, the bias current graphs do not account for the maximum power loading of LEDs. LEDs may only be pulsed using forward currents greater than 25 mA (nominally), when complying with the following constraints allowing operation up to 100 mA or in some cases 50 mA: switching frequency 1 kHz, mark/space ratio 1:10 and 25 °C ambient temperature.

Some manufacturers of super-bright LEDs select devices for uniform luminosity within ±1.6 mcd, specially for use in traffic indicator equipment. To avoid blending effects, the selection is purposely not for maximum brightness as you might have expected. In traffic lights, Indium-Gallium-Nitride LEDs are employed with a luminosity of 'just' 180 mcd. By contrast, LEDs with a luminosity of 9,000 mcd (!) (like the L5-W54S-BS) are preferred for room lighting, solid-state torches and so on.

(020292-1)

Subsonic Microphone

using a low-cost electret insert

by K.-H. Kopp

Commercially-produced subsonic microphones are too expensive for the cost-conscious user. However, as long as a high-precision measuring device is not what is required, you can build such a microphone yourself using a few components from the spares box.

The job of a microphone is to convert a sound into an electrical voltage as faithfully as possible. The conversion happens in two stages: first from sound into mechanical form in the microphone capsule, and then from mechanical into electrical form in the transducer itself. The chief characteristics of a microphone are its directional response and its frequency response. The directional response is only of peripheral interest to us here: it is the frequency response that concerns us more. How can a microphone, which according to its specifications has a frequency response starting at say 50 Hz, be made to respond to infrasound at frequencies below a few Hertz? To answer this question, we first need to go into a little theory.

The frequency response of a microphone is determined jointly by the action of the capsule and the transducer. Here we distinguish between dynamic types, such as the ribbon and moving coil microphones, and electrostatic types, such as the condenser microphone. Microphones employing other types of transducer, such as crystal, carbon or contact microphones, are now only of historical interest.

If we wish to construct an economical and simple microphone, the choice of transducer type is easy: only the electret microphone, which operates on the condenser principle, is worth considering. Many different electret capsules are readily available and they are inexpensive. They require no capsule supply voltage, since the membrane (diaphragm), or electrode, is made of an electret material (generally based on a PTFE film) which has received a permanent electric charge by electron bombardment. In higher-quality electret microphones the back electrode is also made of charged electret material.

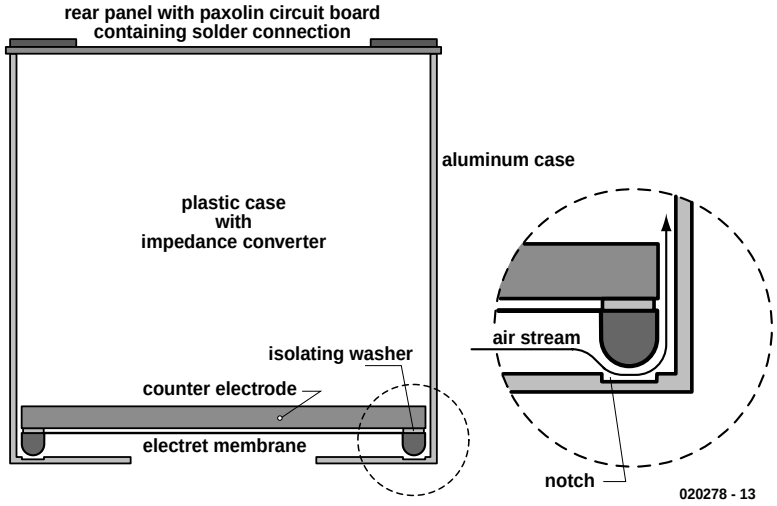


Figure 1. Cross-section through an electret microphone capsule.

Figure 1 shows an electret microphone in cross-section. The electret diaphragm is fitted to the front side, and a short distance behind it is the back electrode. The diaphragm is laid on an insulating ring. The majority of the capsule consists of a plastic body containing a tiny impedance converter (a simple FET stage). The tiny Paxolin circuit board on which the impedance converter is built also serves as the rear panel holding the connection terminals. The whole thing is fitted into a small aluminium tube with rolled edges.

Pressure and pressure gradient transducers

If an electret microphone is examined closely, a very tiny feature will be found that nevertheless has far-reaching consequences for its frequency response: there are notches in the rim of the aluminium enclosure, under the insulating ring. Air movement is possible through these notches between the front and the

rear of the diaphragm (membrane). This changes the capsule from a pressure transducer into a pressure gradient transducer.

Figure 2 shows the main difference. In a pressure transducer the diaphragm moves outwards when the external pressure is low (a trough in the sound wave), and inwards when the pressure is high (a peak in the sound wave). However slowly the air pressure changes, the diaphragm will (at least in theory) always be displaced. In practice pressure transducers are equipped with a capillary hole which allows slow equalisation of pressures as the atmospheric pressure changes. Matters are different with the pressure gradient transducer. Here the displacement is a result of difference in path lengths and hence difference in sound pressure (or gradient) between the front and rear sides of the diaphragm. It is not necessary to express this behaviour in a formula to see that the pressure gradient becomes smaller and smaller as the frequency is reduced. The response of the pressure gradient transducer to very low frequencies is very weak, in stark contrast to the pressure transducer, which in theory can operate down to 0 Hz.

Ingredients

All it is necessary to do to construct a subsonic microphone, then, is to convert the pressure gradient transducer into a pressure transducer — for example, by sealing up the notches with glue. That is fine in theory, but is unfortunately not practicable since the delicate diaphragm will be damaged by the glue. Instead, we can simply replace the enclosure. Since not everyone will have access to a lathe to turn up a sealed made-to-measure enclosure, the author has found that standard components can be misused to make a suitable enclosure.

Take a phono line socket, a stereo jack plug (3.5 mm) and a BNC socket with cable gland, as shown in disassembled form in **Figure 3**. It is important to ensure that the phono socket and the jack plug are made of metal (the BNC plug is only available in metal). Only parts of the three connectors are used: the shells of the phono coupler and jack plug, and the cable gland of the BNC plug.

The shell of the phono socket will later be used to hold the microphone insert after it has been removed from its enclosure. The internal diameter of the shell needs to be at most a few tenths of a millimetre smaller than the external diameter of the disassembled microphone. Hold the shell in a drill chuck and wrap a suitable dowel in grade 400 sandpaper. As in a lathe, the shell is reamed out as far as is necessary, taking care not to damage the thread. Next, cut down the shell of the jack plug, which must have the same diameter as the 'naked' microphone, to about 10 mm, and carefully smooth it off. **Figure 4** shows the individual parts of the new microphone enclosure in the order in which they are assembled.

Now the microphone is prepared. The rolled edge on the rear should be separated from the circuit board using a sharp craft knife. Ensure that you do not touch the microphone aperture, since the diaphragm is very easily damaged. Do not yet remove the microphone: first solder a shielded cable to the solder connections, observing their polarity.

Under cover

You now require a clean room, which can be improvised from a transparent refuse sack. Since dust, grease, dirt and electrical charge will all damage the diaphragm and the internal components of the capsule, earth yourself, slip on thin rubber gloves (without talcum powder!), and proceed to carry out the rest of the work in the clean room.

Hold the capsule vertically by its cable

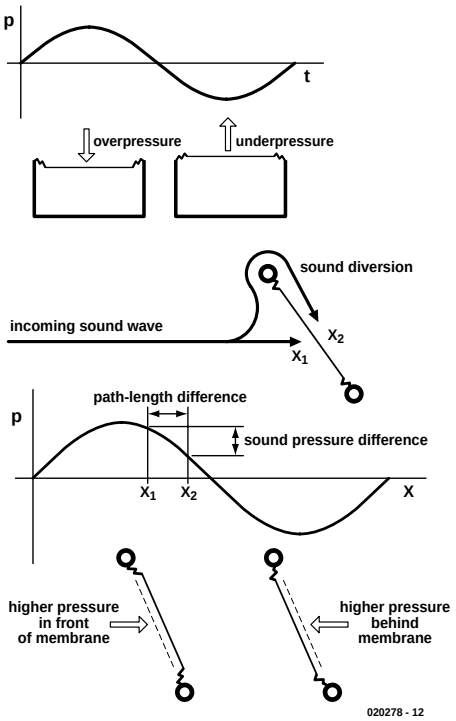


Figure 2. Pressure transducers and pressure gradient transducers compared.

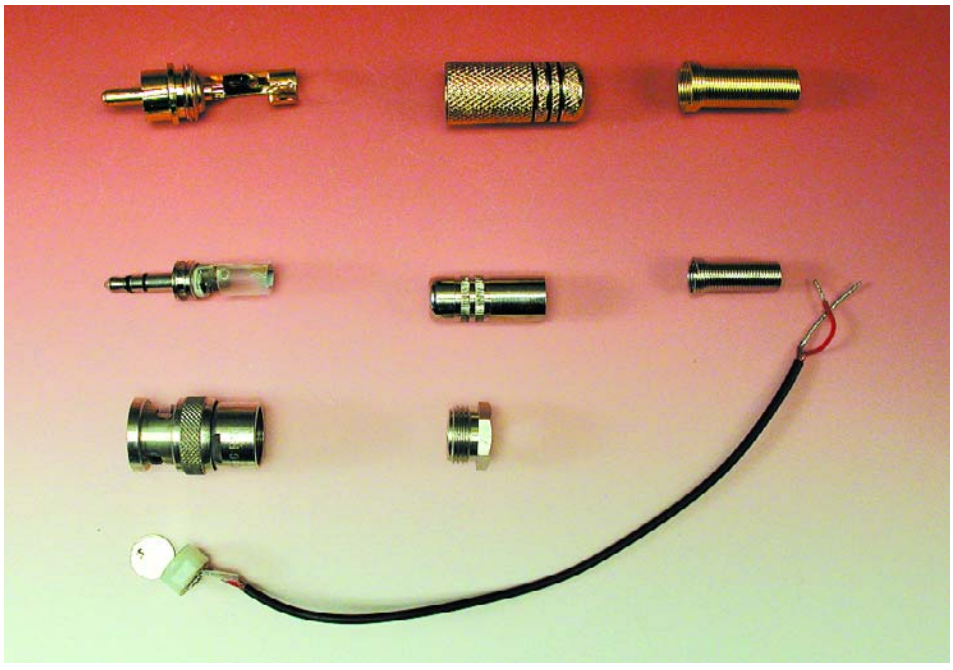


Figure 3. The three connectors used.

and carefully lift off the aluminium enclosure. Then, equally carefully, guide the shell of the phono socket over the capsule without touching the internal parts. The diaphragm insulating ring now lies against the internal edge of the shell which would normally hold the cable strain relief in the phono socket. Fit the sawn-off jack plug shell and fix it using the BNC cable gland (**Figure 5**).

The subsonic microphone, as we can now call it, can now be taken out of the clean room. The enclosure should now be adequately airtight to allow measurements in the subsonic range to be carried out. To make absolutely certain, use a needle to add a drop of solvent-free superglue at the point of contact between the shell and the insulating ring, taking care not to touch the diaphragm.

The open diaphragm is very fragile and must be shielded from dust and contact. To protect it, fit a suitable 20 mm length of plastic sheath over the front and fit a foam cap to the aperture.

Electronic amplification

The output voltage of a condenser microphone is not very high and must be amplified by a large factor before the signal can be processed further. The electronics for the initial impedance conversion, which takes place inside the capsule, requires a power supply which is provided in **Figure 6** by R1 and R2. The AC signal is taken from the microphone via coupling capacitor C3 to the first amplification stage IC1.A. The input impedance is set by R3, and the gain is given by 1+R5/R4. C4 limits the bandwidth of the amplifier and prevents instability. With a microphone voltage of –30 dBu to –70 dBu a gain of 34 is not adequate. A second amplification stage using IC1.B amplifies the signal by a further factor of 3.3 or 30. This gives a total gain of 100 (with S1 in the position shown) or 1000 (with S1 in the other position). This allows any readily-available electret microphone to be used (as long as it is the correct physical size), and the output of the circuit will provide a signal at line levels.

The dual opamp used is a precision device (i.e., with a low input offset voltage) from Maxim with a very low current consumption and so ideal for battery operation. The data sheet can be found at
http://pdfserv.maxim-ic.com/arpdf/MXL1178-MXL1179.pdf

To analyse the subsonic sound, connect the microphone preamplifier to the input of a sound card. A suitable program (with a free experimental version) for analysing oscillations and sounds is *Medusa*, which can be found at

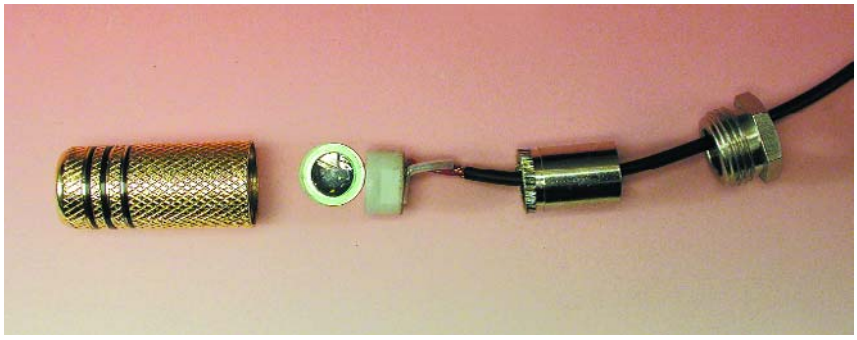


Figure 4. The prepared components.

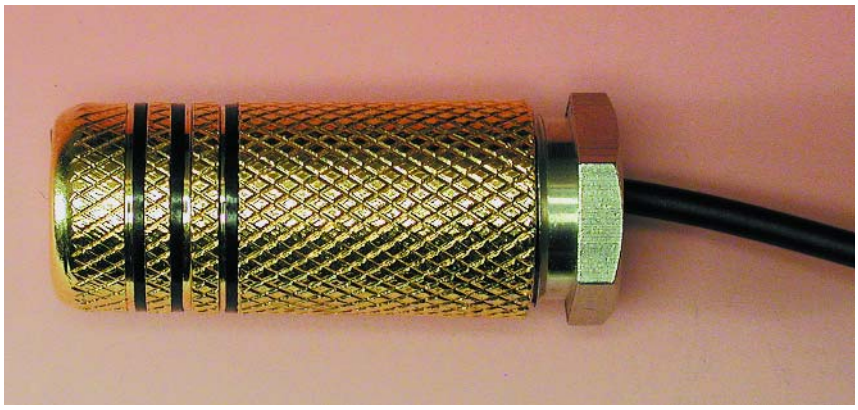


Figure 5. Assembly of the components in the clean room.

www.maschinendynamik.de/index.html

A guide to constructing this microphone can also be found on the Internet at the site of the German interest group for investigating low-frequency sounds (IGZAB) at
http://www.brummt.de

Microphones, microphone amplifiers and sound cards do not exhibit a flat frequency response in the range below 10 Hz. If you are not able to measure the transfer characteristics of the various parts of the

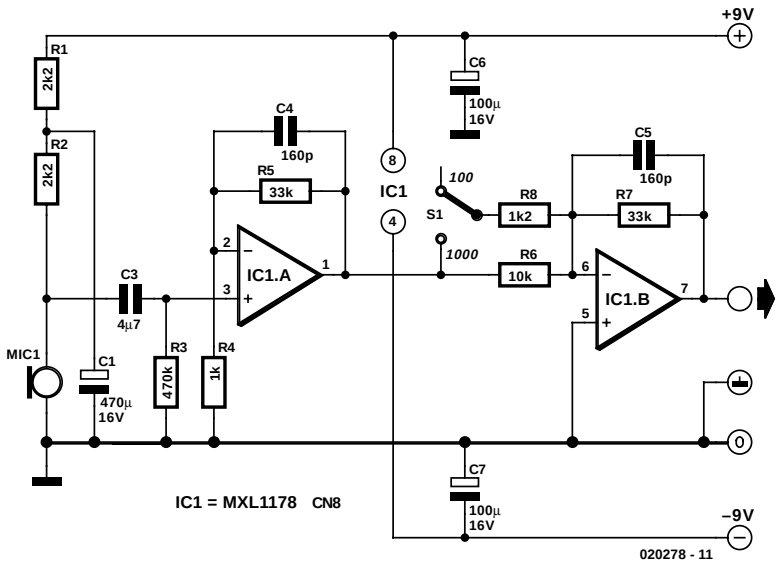


Figure 6. Circuit diagram of the microphone preamplifier.

measurement chain, you will only be able to work with estimated values.

The transfer characteristic of the subsonic microphone is not known and is therefore taken (rather optimistically) as 1. Correction factors for the microphone preamplifier used and for the (Terratec) sound card have been determined as shown in **Table 1**.

Of course, subsonic sounds can be recorded using an A-D converter instead of a sound card. The author used the *Edirol UA-1A USB Audio Interface* from Roland. This gave different correction factors, see **Table 2**. If the two inputs of the Edirol unit are used with two microphones mounted a set distance apart, it is possible to investigate techniques for locating the source of a sound.

(020278-1)

Table 1

Frequency (Hertz)	Correction factors			
	Subsonic microphone	Preamplifier	Terratec soundcard	Overall
10	1	1	1	1
4	1	1	0.62	1.6
2	1	1	0.37	2.7
1	1	1	0.25	4

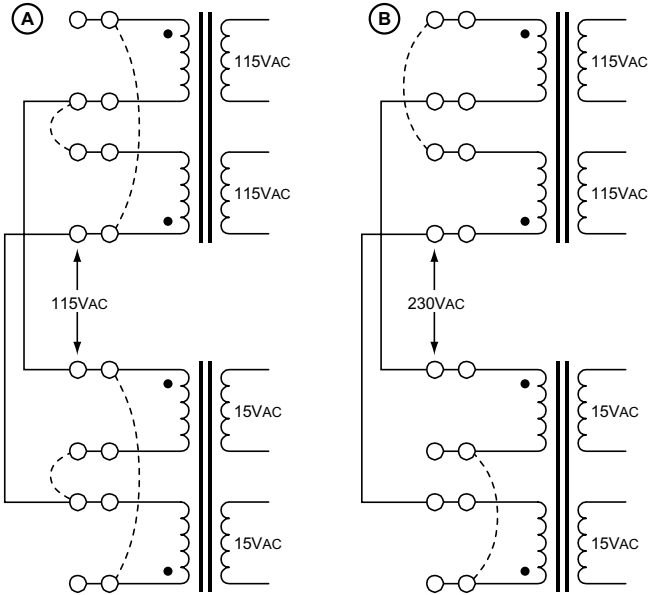
Table 2

Frequency (Hertz)	Correction factors			
	Subsonic microphone	Preamplifier	Edirol UA-1A	Overall
10	1	1	1	1
4	1	1	0.94	1.1
2	1	1	0.85	1.2
1	1	1	0.73	1.4

Tube-Based Crossovers

January 2003, p. 60-65 (020297-1).

A number of errors have crept into the drawings in Figure 3 on page 61 of the article. In Figure 3a, two unnecessary and possibly harmful wire links are shown, while the labels '115 V' and '230 V' in Figures 3a and 3b should have been transposed. Also, all transformer secondaries should have been marked '115 V', not '15 V'. To prevent misunderstandings the corrected drawings are printed here.



Small DC-DC Converters

January 2003, p. 54-58 (020180-1).

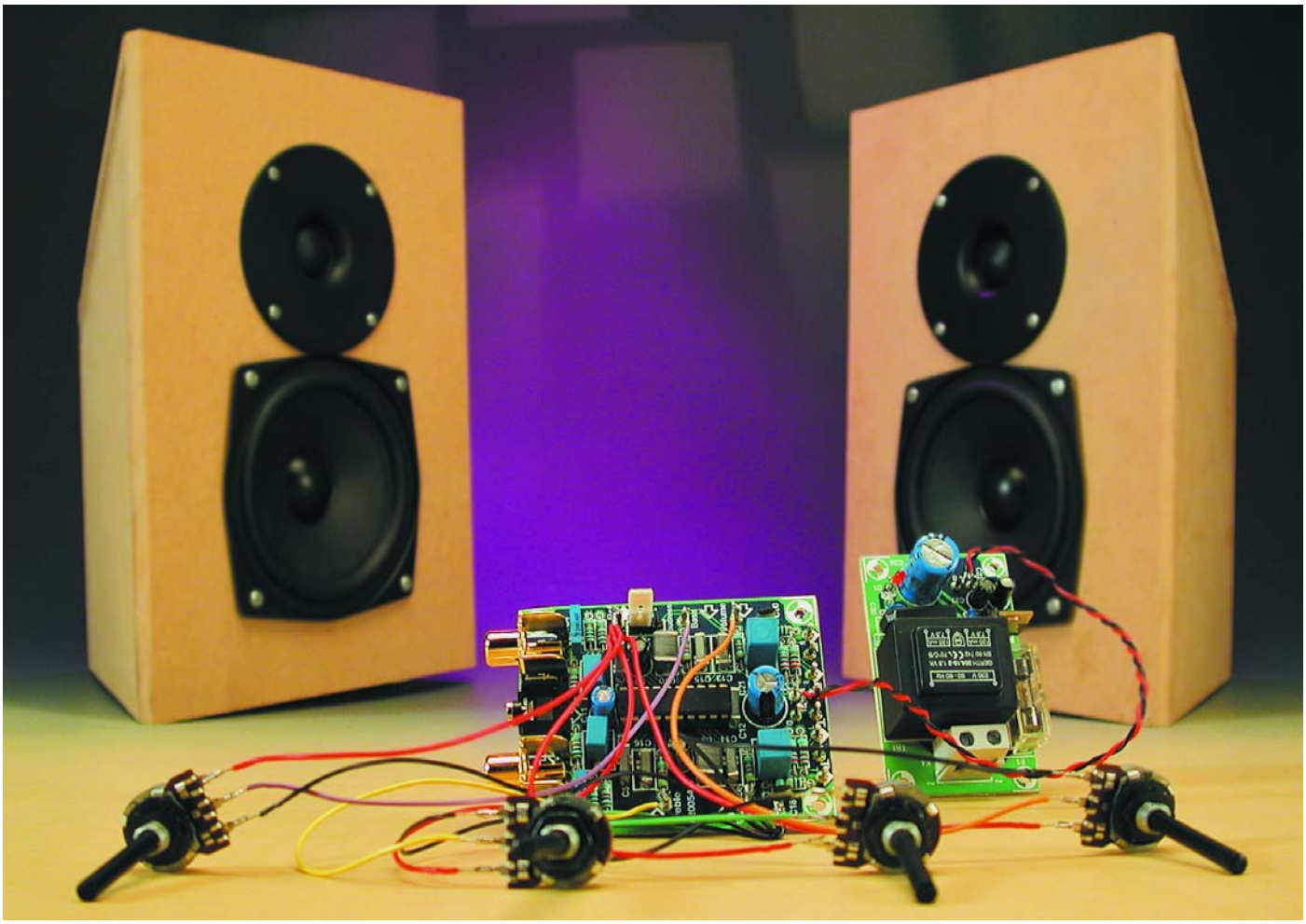
In Figure 3, point 'P' (mentioned in the text) is not indicated. The relevant point is the junction of the collector of T1, choke L1, resistor R1, the cathode of diode D1 and the base of T2.

Single-Chip Tone Control

For use with the multimedia loudspeaker system

Design by T. Giesberts

Although our active loudspeaker system was completed with the recently published subwoofer and associated power supply, we have added a little extra for the enthusiast: a compact (standalone) tone/volume/balance control, complete with a stabilised 15 V supply. Our active system must surely rank amongst the most comprehensive of its type.



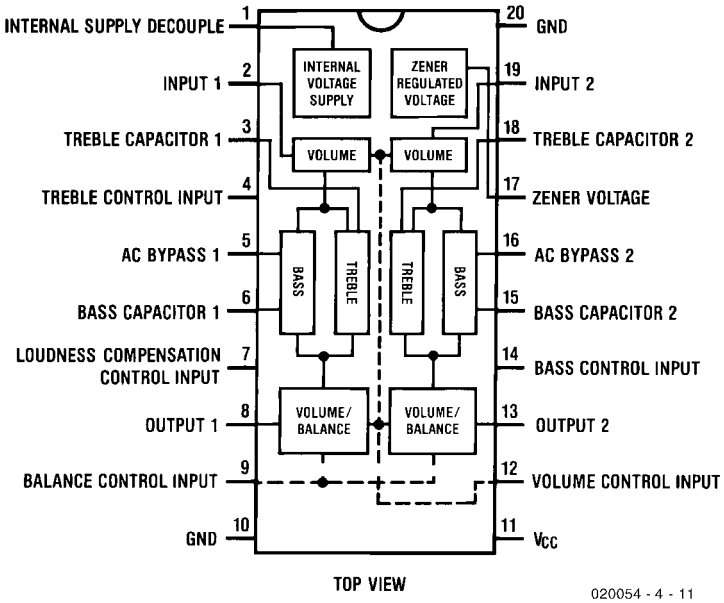


Figure 1. Pin-out and block diagram of the LM1036 (courtesy National Semiconductor).

Something like a tone control is of course not essential for use with most PC loudspeakers. But when the multimedia system is used to play back music then this is a useful extra. Besides, publishing this circuit gives us the satisfaction of knowing that our active system is now really complete.

We've tried to give the tone control as much universal appeal as possible. The circuit can therefore be used on its own as an addition to an existing HiFi system, but at the same time complements our active loudspeaker system as well as possible.

This immediately rules out the use of a discrete circuit. As far as quality is concerned, a well-designed discrete circuit still has the edge on an integrated solution, but the size of such a circuit wouldn't fit in with the rest of the system. For this reason, just as for the amplifiers in the satellites and subwoofer, we searched for a single-chip solution.

The LM1036

In the past integrated (pre)amplifiers were badly thought of in HiFi circles, although nowadays the quality of these chips leaves little to be desired. Since there has been an increase in demand for better quality Mini systems and car radios, this area has seen an increase in the availability of good quality ICs.

We finally decided on the LM1036 made by National Semiconductor, a 20-pin DIL IC that

Measurements

(All outputs have a 10 kΩ load)

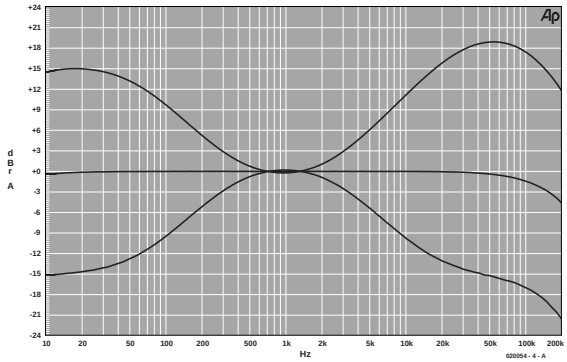
Bass control range (volume –40 dB)	+15 dB / –15dB (20 Hz)
Treble control range (volume –40 dB)	+16 dB / –13dB (20 kHz)
Bandwidth	16 Hz to 70 kHz
Signal to noise ratio	78 dBA (250 mV in)
THD+N (B = 22kHz, 250 mV output)	0.04 % (at maximum gain)
Maximum gain	0.6 dB (≈ 1.07 x)
Maximum input level (250 mV output)	1.93 V (14.43 V supply,
THD+N = 0.3 %)	
Maximum output level	0.75 V (THD+N = 0.3 %)
Supply voltage range	9 to 16 V
Supply current	46 mA

Response curves:

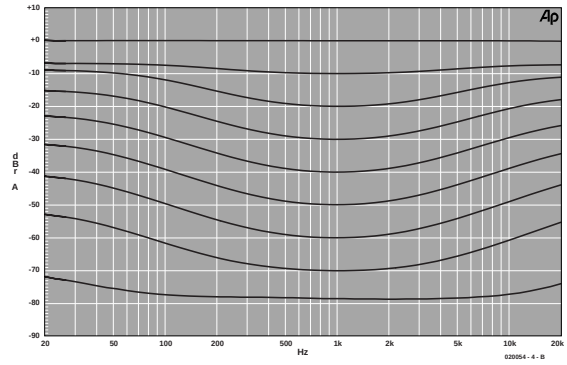
Graph A shows the frequency response with the bass and treble controls at their extreme settings, with the middle curve showing the response when the tone controls are at their central setting. The volume control was set to –40 dB. We have deliberately shown the response between 10 Hz and 200 kHz, so you can see what happens outside the audio spectrum.

The second graph (B) demonstrates the effect of the loudness compensation. We measured the output level between 20 Hz and 20 kHz at nine different volume levels, with –10 dB (at 1 kHz) between each step. We've used a slightly larger input signal (700 mV) and a band-pass filter so that the range of the volume control could be seen more clearly. The top curve was measured with the volume control at its maximum. The loudness compensation has the strongest effect between –50 to –60 dB. At the bottom curve (0 V control voltage) the output consists mainly of noise and that is the reason why the effect is no longer very noticeable.

A



B



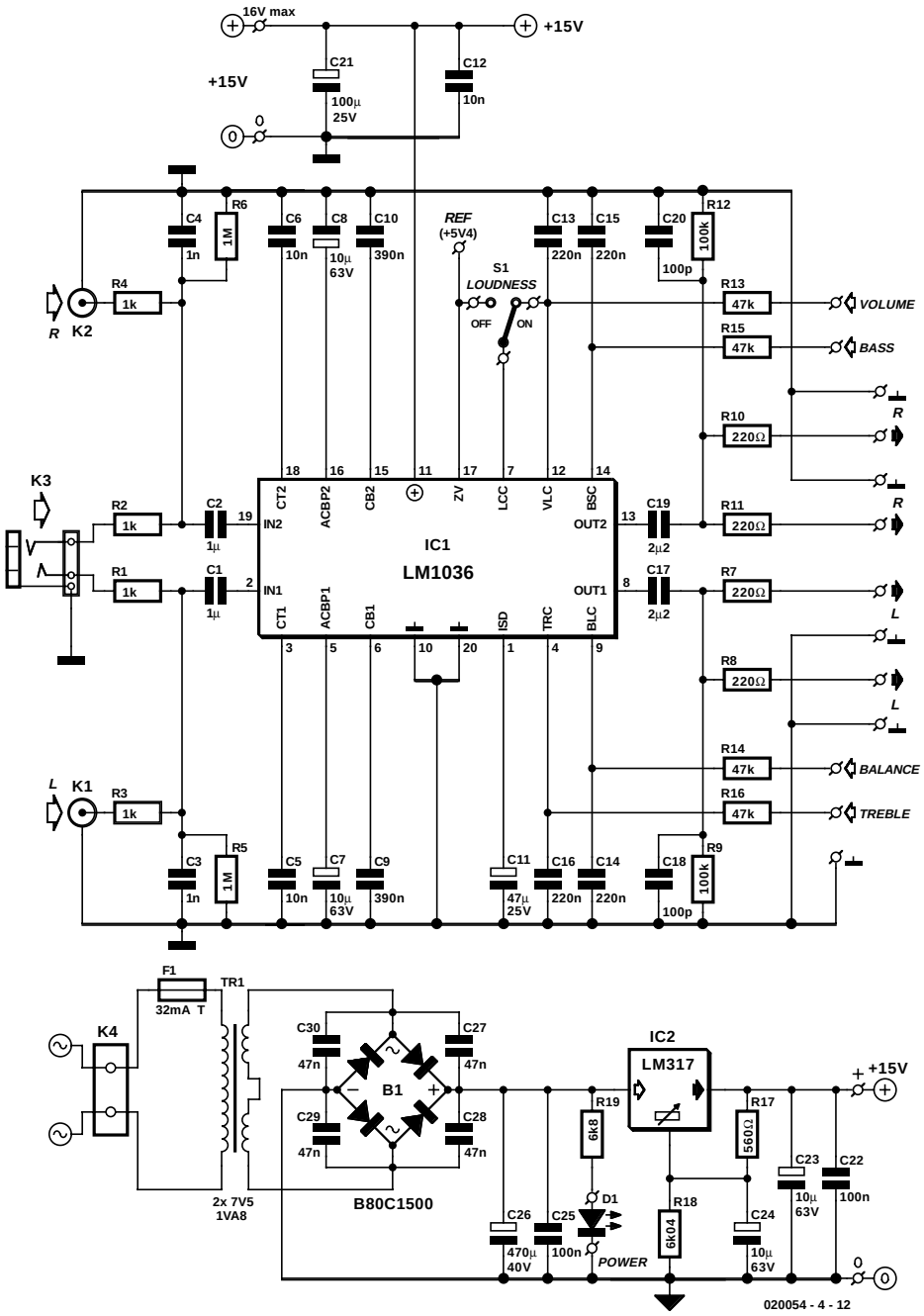


Figure 2. Since all control functions are integrated in IC1, the circuit is an example of simplicity.

contains, with the exception of a handful of external components, a complete stereo volume/tone/balance circuit, and furthermore has a separately switched loudness control. All functions are controlled via DC voltages, so there is no need to worry that mains hum is picked up by the cables between the potentiometers and the PCB. The IC can be used with a supply voltage between 9 and 16 V (a car battery could therefore be used). It has a large volume control range of 75 dB as well as a large tone control range of ±15 dB.

Although the IC doesn't claim to have a high-end performance, its quality is more than sufficient here. The figures given by the manufacturer state that the channel separation is 75 dB, the total harmonic distortion (THD) is 0.06% and the signal to noise ratio is 80 dB. It can be seen that these figures aren't an overstatement when you refer to the section at the end, which shows some measurements and specifications of our prototype.

Figure 1 shows the pin-out and a simplified block diagram of the LM1036. For those of you who are interested, the complete datasheet can be found at www.national.com/pf/LM/LM1036.html

Simplicity itself

The complete circuit diagram in **Figure 2** illustrates how few parts are required for a tone control built around the LM1036. Apart from the IC, there are only a handful of resistors and capacitors, a few input and output connectors and four potentiometers, and that is all.

To make the circuit more versatile we've added a mini jack input (K3) to the usual phono inputs (K1, K2). Only one of these types of input should be used, otherwise the signals will be mixed by R1/R3 and R2/R4 and both sounds will be heard simultaneously.

Each of the left and right channel outputs has a twin set of solder pins on the PCB, making it easier to connect the active 2-way loudspeakers and subwoofer. You are free to choose any type of connector for this.

Four control voltages are required for controlling the volume, treble, bass and balance. These are obtained from four single-ganged linear potentiometers (P1-P4). The ends of these are connected to the internal 5.4 V zener reference (pin 17) and ground, the wipers are connected to VL, BL, BS and TR (pins 4, 9, 12 and 14).

Circuit details

Each control input is decoupled by an RC network (R13-R16, C13-C16). The treble and bass control ranges are set using only a single capacitor per channel. These are C5 and C6 for the high frequencies, and C9 and C10 for the low frequencies.

Another three electrolytic capacitors are required to decouple internally generated voltages (C7, C8 and C11). C1 and C2 decouple the inputs from DC offsets. At the minimum input impedance of the IC (20 kΩ) the roll-off frequency at the input is about 8 Hz. The capacitors at the output (C17 and C19) have a larger value than C1 and C2 because the total load impedance of the active 2-way system and subwoofer is about 6.4 kΩ. C18 and C20 have been

added to suppress internal HF oscillations when the output is only slightly loaded.

As mentioned earlier, the IC contains a loudness compensation stage, which gives an increased amplification to lower and higher frequencies at lower volume settings (refer to response curve B). When the 'loudness compensation control' input (pin 7) is connected to the internal zener reference (pin 17), the effect is turned off. The function is turned on by connecting pin 7 to the 'volume control' input (pin 12). We have included a switch for this in the circuit, because we assumed that most constructors would like the facility to switch the loudness function on or off at will. It is also possible to place a 3-pin header on the PCB if you want to keep things simple and have a jumper to set the effect on or off permanently.

Power supply

Although the nominal supply voltage for the LM1036 is 12 V, we decided to use 15 V because the IC can deal with input signals up to 2 V_{rms} at this supply voltage. To spare you the effort of searching for a suitable stabilised 15 V power supply, we've added such a circuit to the design.

We used an LM317 (IC2) for the voltage regulator, because that has a better supply ripple suppression (typ. 80 dB) than standard 78xx regulators. R17 and R18 set the output voltage to 15 V. However, this only applies for the typical values (1.25 V between output and control input, 50 µA adjust current). Due to unavoidable tolerances we found that the output voltage in our prototype was only 14.43 V. If you want to obtain a precise 15 V output you should vary R18 a little (increase its value for a higher output voltage).

The rest of the supply is a standard circuit: a smoothing capacitor (C26) and HF suppression (C25), a bridge rectifier (B1) with HF suppression (C27-C30), a transformer and mains fuse. LED D1 is used as a power indicator.

PCB

Figure 3 shows the PCB for the tone control circuit. The power supply section is on the right hand side of the board; this part can be cut from the main PCB if you want to mount it elsewhere in the enclosure. A larger physical separation between the signal processing and supply sections always reduces any possible interference.

The usual rule of mounting components from 'low to high' applies particularly to the power supply section. Should you mount the transformer and bridge rectifier first, you'll find that it becomes very tricky to subsequently solder C27, C28 and C29 into place. A 1.8 VA transformer made by Gerth is given in the parts list, but this has a standard footprint, so other common 1.5 VA types should fit as well.

The construction of the tone control section shouldn't require much time. There aren't that many components and the PCB is well organised. All connections are clearly labelled and positioned logically: inputs on one side, outputs on the opposite side and the connections for the four potentiometers (Treble, Balance, Volume, Bass, Ground (⊥) and Ref) are evenly divided over the remaining sides.

We have already mentioned loudness switch S1. In our prototype we used a pin header and jumper for this. The supply connections have been positioned as close as possible to decoupling capacitors C12/C21 and the outputs. **Figure 4** shows our prototype board.

The choice of enclosure is left up to the reader. There are many enclosures available that can neatly accommodate both PCBs and the four potentiometers. Remember to use a properly isolated mains cable and use a strain relief grommet where the cable enters the enclosure so that terminal block K4 can't suf-

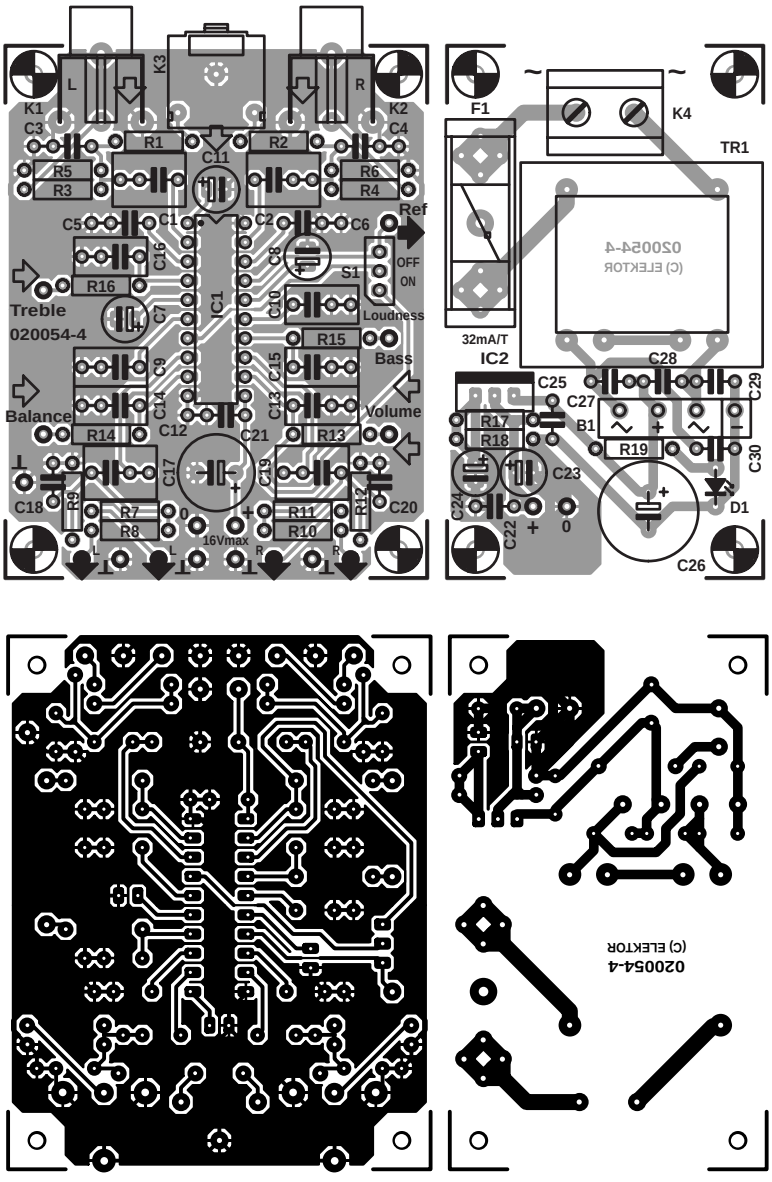


Figure 3. Track and component layout of the PCB. The supply section can be separated from the rest (board available ready-made).

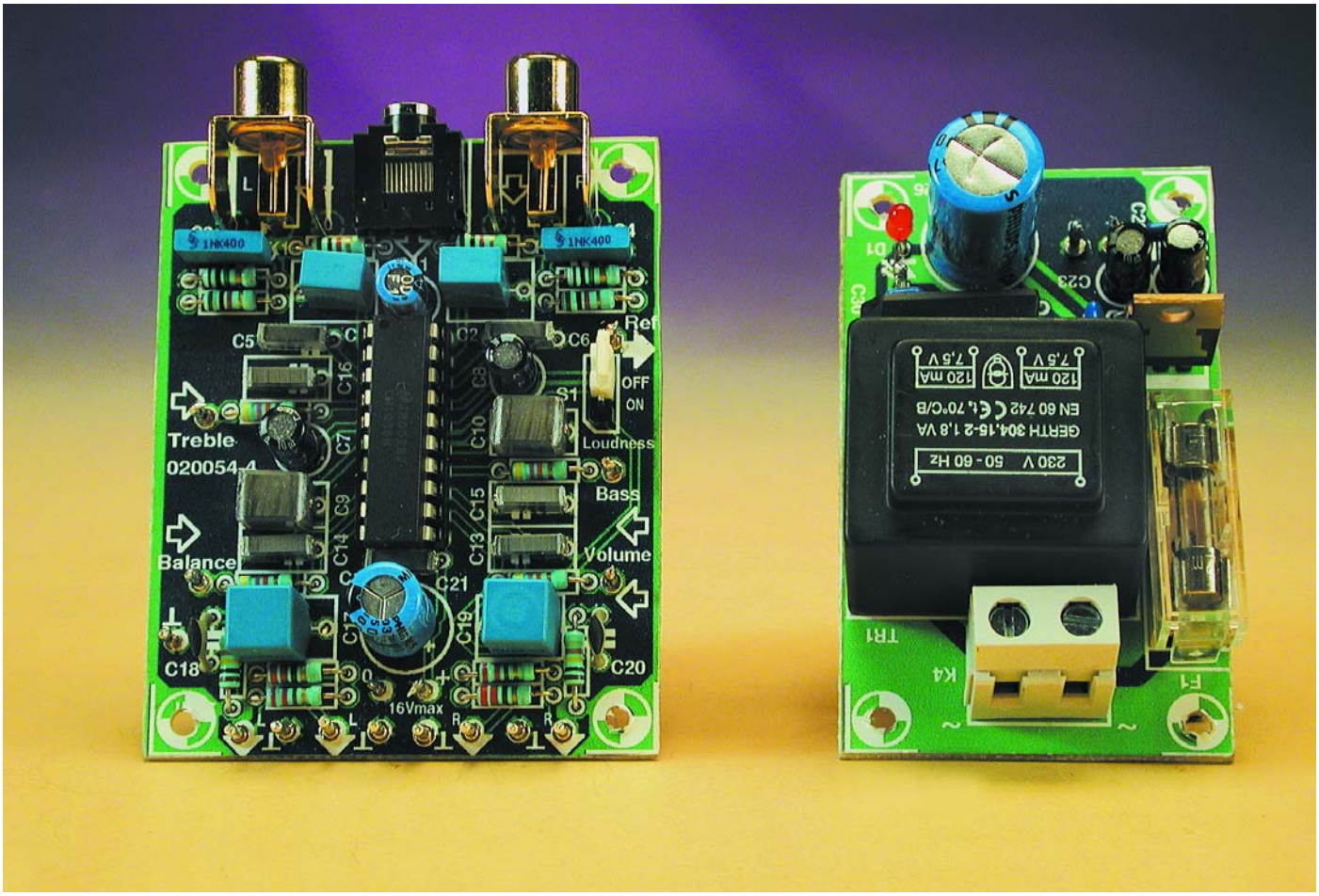


Figure 4. Populating the board should be a fairly easy task.

fer from any mechanical strain. The electrical safety page, which is published regularly in Elektor Electronics, contains many practical guidelines on working with mains voltages. When the circuit has been housed you should apply an identifying label, showing the PCB number and the value of the mains fuse.

A big advantage of this circuit is that the four control inputs are driven by a DC voltage, meaning that the audio signal does not go to and from the potentiometers. In practice this has the benefit that the wiring to the potentiometers doesn't require shielded cable, but that ordinary flexible isolated connecting cables can be used. The wiring is therefore not critical and there is no need to worry about mains hum!

(020054-4)

COMPONENTS LIST

Resistors:
R1-R4 = 1kΩ
R5,R6 = 1MΩ
R7,R8,R10,R11 = 220Ω
R9,R12 = 100kΩ
R13-R16 = 47kΩ
R17 = 560Ω
R18 = 6kΩ04
R19 = 6kΩ8
P1-P4 = 47kΩ linear (mono)

Capacitors:
C1,C2 = 1µF MKT, lead pitch 5 or 7.5mm
C3,C4 = 1nF
C5,C6,C12 = 10nF
C7,C8,C23,C24 = 10µF 63V radial
C9,C10 = 390nF
C11 = 47µF 25V radial
C13-C16 = 220nF
C17,C19 = 2µF2 MKT (metallised plastic), lead pitch 5 or 7.5mm
C18,C20 = 100pF
C21 = 100µF 25V radial
C22,C25 = 100nF ceramic
C26 = 470µF 40V radial
C27-C30 = 47nF ceramic

Semiconductors:
B1 = B80C1500 in rectangular case (80V piv, 1.5A) (~+~–)
D1 = LED, high-efficiency
IC1 = LM1036N (National Semiconductor)
IC2 = LM317 (TO-220 case)

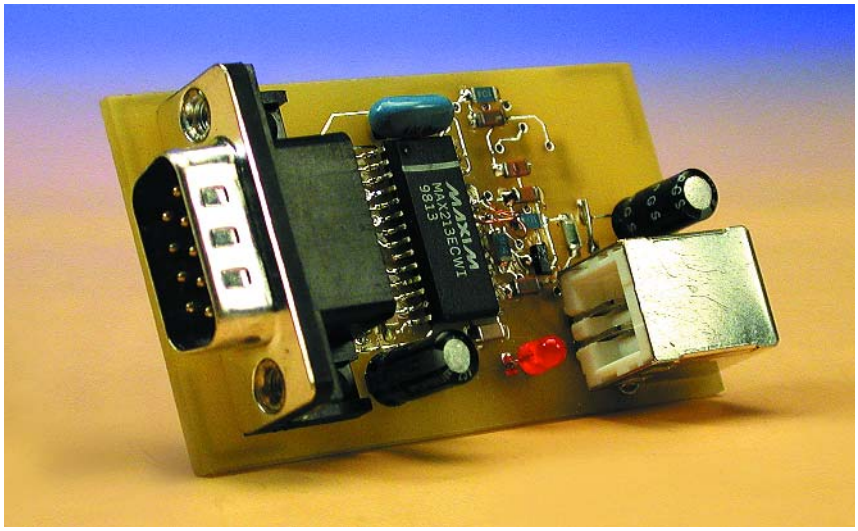
Miscellaneous:
K1,K2 = cinch socket, PCB mount, e.g., T-709G (Monacor/Monarch)
K3 = 3.5-mm stereo jack socket, PCB mount
K4 = 2-way PCB terminal block, lead pitch 7.5mm
S1 = switch, 1 changeover contact
F1 = fuse, 32 mA/T (time lag) with PCB mount holder
TR1 = mains transformer, secondary 2 x 7.5V at 1.8VA (e.g., Gerth type 304.15-2)
PCB, order code **020054-4** (see Readers Services page)

USB-RS232 Interface

A compact solution for missing ports

Design by L. Lemmens

Thanks to a special integrated circuit from FTDI Chip, computer peripherals with an RS232 interface are easily connected to a USB port. This simple solution is ideal if a peripheral does not have a USB port, your notebook PC has no free RS232 port available, or none at all!



After a slow and faltering start, the USB port has become commonplace on PCs, to the extent that the latest GHz machines have just one RS232 port left, or none at all. The compact USB-RS232 interface described in this article allows your good old RS232 peripherals (printer, programmer system, etc.) to be hooked up to a USB port. The free driver programs for Win98/ME/2000/XP, Linux and Apple Macintosh make the interface virtually transparent, enabling the USB port to behave like a regular COM interface. The driver and the conversion chip from Glasgow-based FTDI Chip allow a full serial data link to be set up on a 9-way RS232 connector, including all handshaking signals. The UK representative for FTDI Chip is Alpha Micro Components Ltd. in Basingstoke (www.alphamicro.net). International distributors for FTDI Chip products may be found on the 'Sales Network' page which may be accessed via www.ftdichip.com. FTDI Chip has representatives in most parts of the world and hosts a very useful website.

Function and architecture

Although it is not necessary to know all the ins and outs of the converter chip if you just want to use the circuit we're about to describe, it is still useful to have an idea of what is going on inside the black box.

The simplified block diagram of the FT232AM is shown in **Figure 1** and the pin assignment, in **Figure 2**. Like its family member the FT245AM, the FTDI Chip FT232AM is essentially a serial USB FIFO (first in, first out register), which is controlled by the computer by way of a virtual COM port. The difference between these two components is that the FT232AM contains a UART which in turn comprises an RS232 interface (using TTL levels). The FT245AM, on the other hand, features a 8-bit interface with handshake lines that allow direct access to the on-chip FIFO. This makes the '245 IC particularly interesting if an existing microcontroller system is to be upgraded with a serial interface. Note, however, that you will need to provide the necessary software 'glue' yourself!

At the side of the USB, the two chips are identical and not surprisingly they contain the same drivers in the relevant section. At the input side, an USB Transceiver forms the link with the USB cable and its two signal wires called D+ and D– for the USB 1.1 Full-Speed mode. The 3.3-V reference voltage needed for the USB Transceiver is supplied by an internal low-drop voltage regulator whose output is available at pin 6 of the chip. This is done not only to enable the reference voltage to be applied to an external decoupling capacitor but also to allow the USB Full-Speed mode to be defined. With reference to the circuit diagram shown in **Figure 3**, that is achieved using resistor R6 which pulls the D+

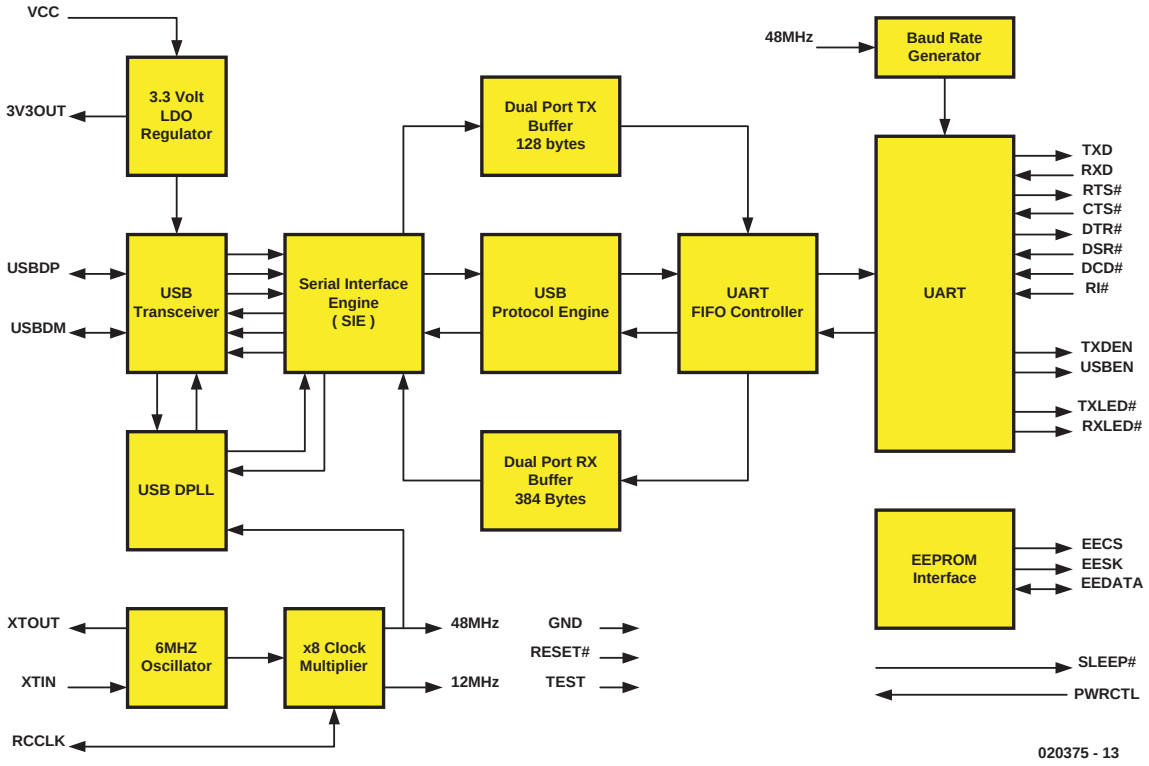


Figure 1. Simplified block diagram of the USB/RS232 converter type FT232AM (courtesy FTDI Chip).

line to +3.3 V. This level causes the USB Host (i.e., the USB controller in the PC) to recognise our interface as a Full-Speed device and arrange for the appropriate addressing. In the case of a Low-Speed device, the D– line is held at +3.3 V with the aid of a resistor. Behind the USB Transceiver we find a functional block identified as 'Serial Interface Engine' which handles the parallel-to-serial and serial-to-parallel conversion of USB data. Next, the 'USB Control Engine' processes the USB control information and looks after the communication with the USB Host Controller (in accordance with the USB Low Level protocol), as well as the commands that define the UART's functional parameters.

Buffers for 'receive' and 'transmit' ('Dual-Port TX Buffer' with 128 bytes capacity and a 384-byte 'Dual-Port RX Buffer') arrange for the exchange of data in both directions (between Serial Interface Engine and the UART registers). The block identified as 'UART FIFO Controller' is responsible for the exchange process between the two buffers and the transmit and receive registers of the UART.

Functionally, the 'UART' proper is not unlike the one found in the PC. Its task is to supply all relevant signals to the RS232 interface and in addition, for RS422 and RS485.

The 'Baud Rate Generator' allows the serial data speed to be set between 300 bits/s and 2 Mbits/s (actually, up to 920 kbits/s for RS232 and 2 Mbits/s for RS422/485).

The 'EEPROM Interface' on board the FT232AM chip is intended for the connection of an external 93C46 EEPROM chip. Although the FT232AM will work happily without the addition of a non-volatile data memory, the interface will then 'report' as a standard serial device. The small EEPROM allows specific data like manufacturer and product identifier

FTDI Chip contact details

Future Technology Devices International Ltd.,
St. George's Studios, 93-97 St George's Road, Glasgow G3 6JA, United Kingdom.
Tel.: (+44) (0)141 353 2565, Fax: (+44) (0)141 353 2656.
URL: www.ftdichip.com

UK distributor:
Alpha Micro Components Ltd, Springfield House, Cranes Road,
Sherbourne St. John, Basingstoke, Hants RG24 9LJ, United Kingdom.
Tel : (+44) (0)1256 851 770, Fax: (+44) (0)1256 851 771.
URL: www.alphamicro.net

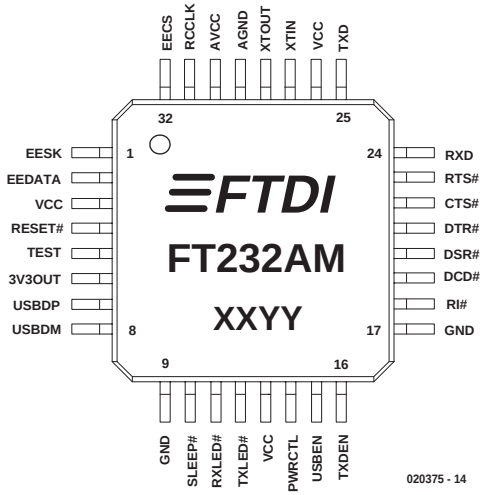


Figure 2. Pin connections of the FT232AM in QFP case (7 x 7 mm) (courtesy FTDI Chip).

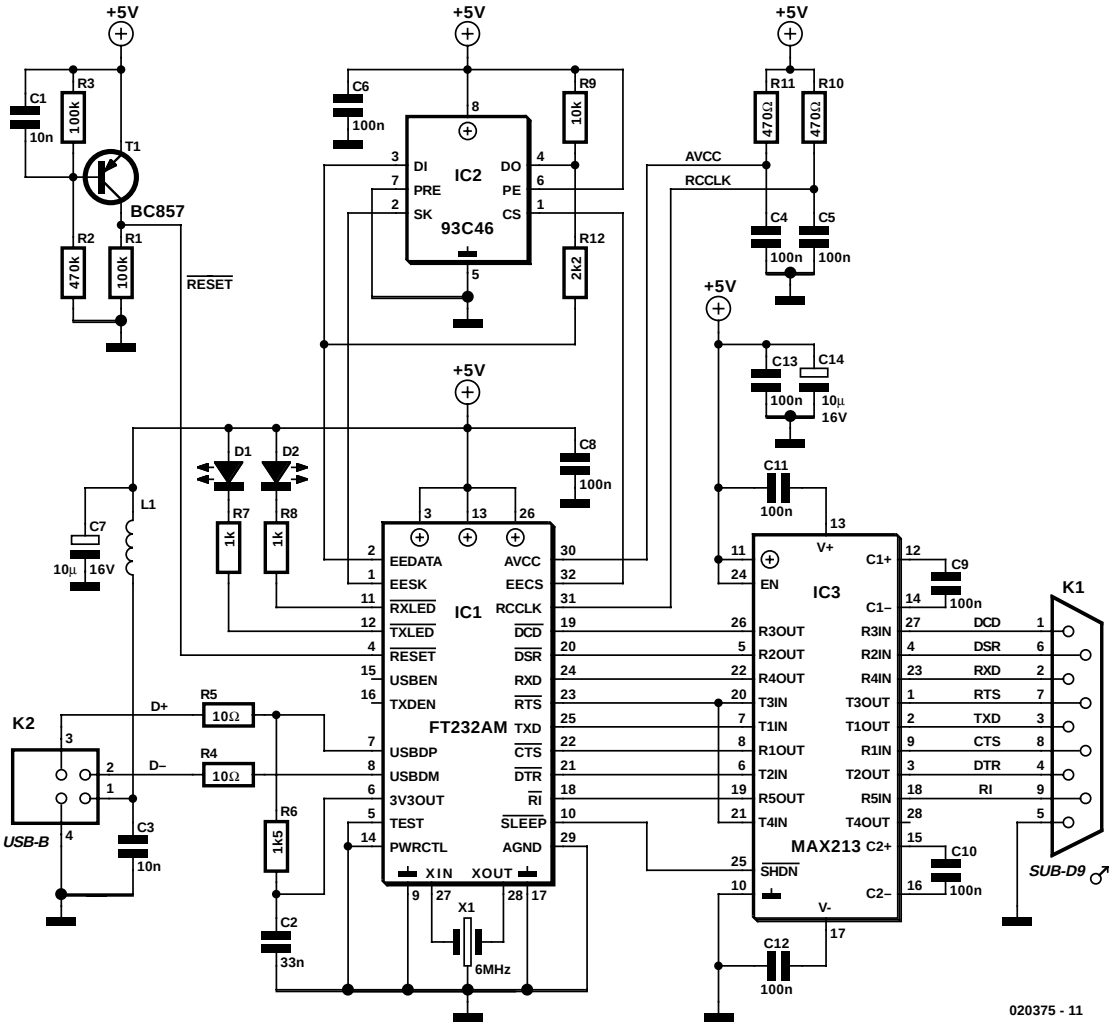


Figure 3. Circuit diagram of the USB/RS232 Interface.

codes (VID, vendor ID; PID, product ID), serial numbers, etc., to be permanently stored and made known to the operating system. Note that the EEPROM is obligatory when several USB/RS232 converters using FTDI chips are connected to a PC. This is because the drivers will only install virtual COM ports for converters with unique serial numbers. Without the serial number (i.e., without the EEPROM) only one virtual COM port can be installed.

Circuit diagram

The circuit diagram shown in Figure 3 looks rather uncomplicated. In the top left-hand corner we find the power-up reset circuit comprising T1 and the customary R-C network. Right beside it is the (optional) EEPROM. In the next lower 'row' we find, from the left to the right, the USB-B connection, the FT232AM, a MAX213 and finally the RS232 header.

The circuit receives its +5 V supply voltage from the PC via pin 1 of the USB connector (K2). A sufficient amount of supply noise decoupling is afforded by a small choke (L1) and capacitor (C7). In addition all integrated circuits have their own supply decoupling.

As already mentioned, resistor R6 pulls the USB D+ line to +3.3 V, in order to tell the USB host that the interface is a 'Full-Speed' device. The same resistor also triggers the recognition of an USB device when the interface cable is plugged into the USB port on the PC or on a hub.

Very conveniently, the FT232AM features two LED driver outputs that allow active data transmission (D1) and reception (D2) to be visualised.

Although the two R-C networks R11-C4 and R10-C5 are identical in value, their functions are quite different. The combination R10-C5 at the RCCLK pin is a timing network to ensure clock stability when the FT232AM wakes up from Sleep mode while booting. The other R-C combination, R11-C4, only decouples the voltage at the AVCC (analogue supply) pin which powers the internal 8x clock multiplier.

The MAX213 and its external charge pump capacitors only serve to convert the 5-V signals at the RS232 side of the FT232AM into true RS232 signals (i.e., having a polarized swing). Normally, that would mean approximately ±12 V, but in practice only ±8 V is achieved, with a maximum of up to ±10 V.

Circuit board

Even if the circuit diagram is uncluttered and fairly straightforward, that does not necessarily apply to the printed circuit board (**Figure 4**). The main reason for the discrepancy is found in the use of SMD parts. Also, the PCB is double-sided and

COMPONENTS LIST

All resistors and capacitors SMD shape 1206

Resistors:
R1,R3,R10 = 100kΩ
R2 = 470kΩ
R4,R5 = 10Ω
R6 = 1kΩ5
R7,R8 = 1kΩ
R9 = 10kΩ
R11 = 470Ω
R12 = 2kΩ2

Capacitors:
C1,C3 = 10nF
C2 = 33nF
C4,C5,C6,C8-C13 = 100nF
C7,C14 = 10µF 16V radial

Inductors:
L1 = BLM31A601S (Murata) (e.g., Farnell # 581-094)

Semiconductors:
D1,D2 = LED, 3mm dia.
T1 = BC857
IC1 = FT232AM or FT232BM (FTDI Chip order code FT8U232AM)
IC2 = 93C46 (optional)
IC3 = MAX213ECWI

Miscellaneous:
K1 = 9-way sub-D plug (male), PCB mount, angled pins
K2 = USB connector, type B, PCB mount
X1 = 6MHz ceramic resonator

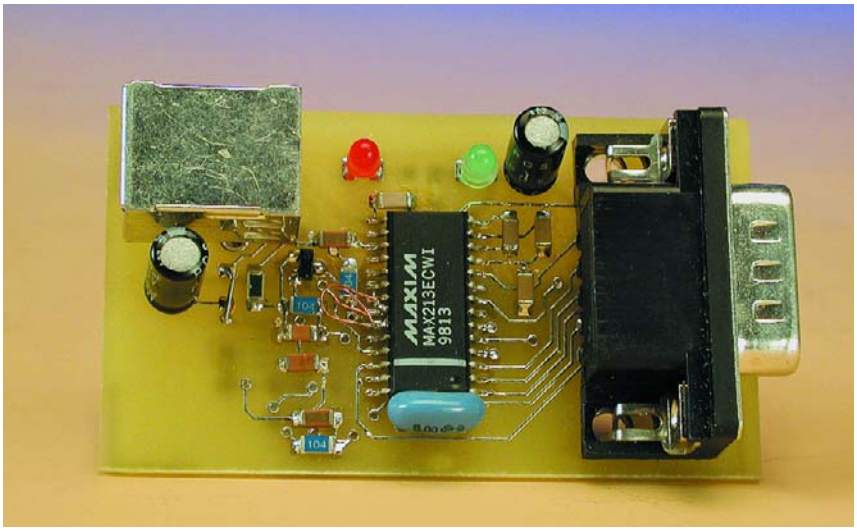


Figure 5. Our finished and fully working prototype board seen from above…

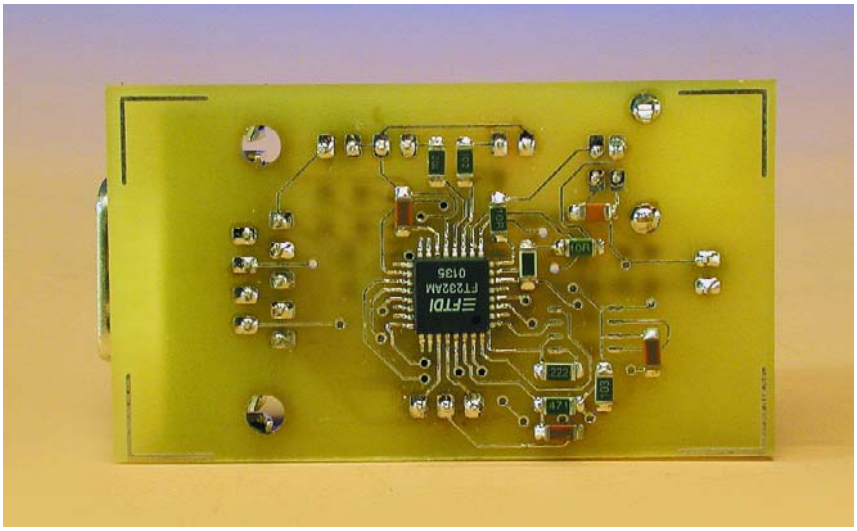


Figure 6. … and from below.

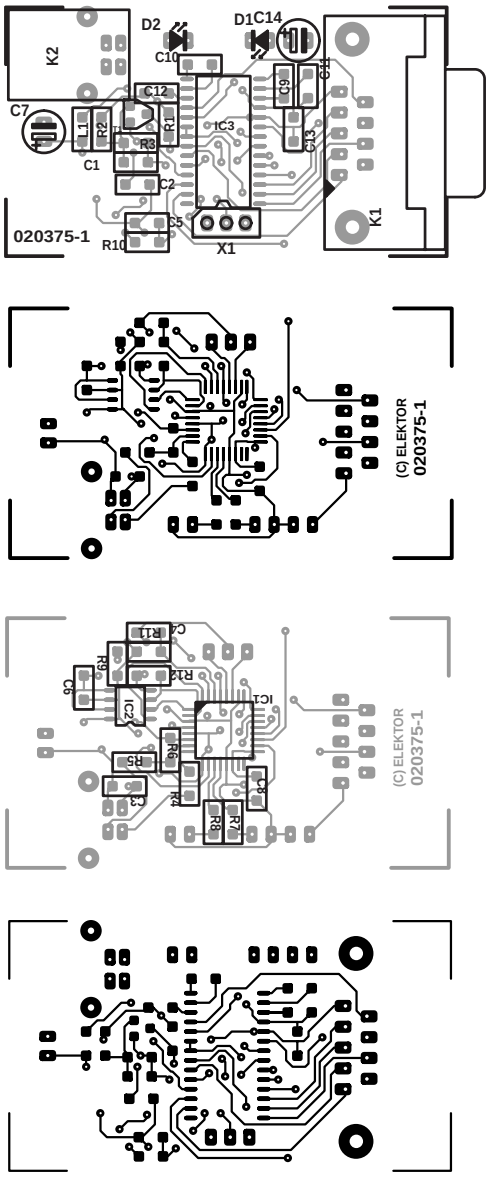


Figure 4. Copper track layout and component overlay of the PCB designed for the interface.

through-plated, and has components fitted at both sides! Those of you who have experience in dealing with SMDs will not be deterred by the extremely compact board layout. Fortunately, for those with less know-how in this area, *Elektor Electronics* recently published a number of useful articles on the subject of SMD soldering (see 'References' at the end of this article). Beginners are advised to prepare themselves by first reading the articles and then acquire practical experience from some (defective) boards and SMD parts to prevent disappointment with the present project.

The bare board is first populated with the SMDs and then with the 'leaded' components and sockets. One of the sockets is a USB Type 'B', for which the pinout is shown in **Figure**

7. The other variant of the socket, called USB 'A', is always at the PC or hub side, while type 'B' should always be inside the equipment reporting to the PC. Via the USB cable, the type 'A' socket supplies current to the 'B' socket at the equipment side. In our case, this current is used to power the USB-RS232 converter board. USB cables always contain 1-to-1 wire connections.

Ready-made modules

Those of you who dislike DIY constructing on PC boards, and SMDs in particular, will be glad to know that FTDI chip also supply ready-made USB-RS232 interface modules under part number DLP-USB232M, see their website for further details. These modules have all connections brought out to pins of a standard 0.6-inch wide DIP plug.

Software

Before connecting your circuit to the PC's USB port, give your soldering work a last, thorough, visual inspection, for which a magnifying glass will prove very useful.

Next, download the necessary drivers from the FTDI Chip website. Drivers are available for all popular Windows versions, as well as for the Macintosh and Linux platforms. Regarding the Windows drivers, versions are available with and without PNP (Plug & Play) support. The latter are called 'non-PNP'. The difference is small but essential. The drivers with PNP support should only be used when the peripheral connected to the PC via the USB/RS232 interface also installed its drivers using Windows PNP. In other words, in case of doubt, resort to the non-PNP drivers first to prevent problems. Typical problems you may encounter with the PNP supporting drivers include slow booting and erroneous identification of an USB/RS232 converter as a 'pointer device' which will result in failure of the mouse to operate as expected! For Windows XP, there's an additional XPNPNP tool that allows the Plug & Play function to be switched off for serial FTDI interfaces.

With the right driver securely stored in a subdirectory , you may start the installation process by connecting the USB/RS232 interface to a USB port on the PC. After a short delay, Windows will report that a USB device has been found. If there is no reference to a USB device, then there is a problem on our little board. In some cases, all you have to do is unplug the USB connector, wait a few seconds and insert it again, so try this first. When everything goes well, all you have to do is browse the system and click on FTDIBUS.INF to make the rest of the installation proceed.

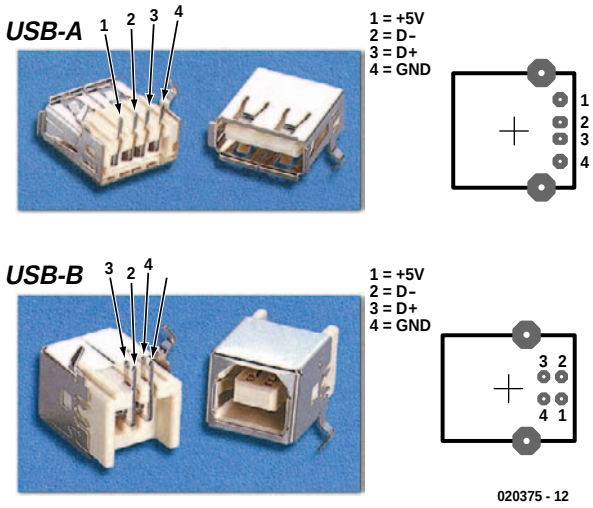


Figure 7. USB-A and USB-B socket pinouts.

Once finished, you will have a new serial port on your system, just check it using Start → Settings → Control Panel → System → Device Manager. If you open the 'Ports (COM & LPT)' folder you should see a new 'USB Serial Port (COMx)' as illustrated in **Figure 8a**.

During the installation, two drivers are actually installed that are linked with one another. One of these provides the virtual COM port you just found as the new device in the Device Manager. The other driver

Installation

As is well known, USB devices may be 'hot plugged' to the PC. The operating system will recognise the interface and request the associated driver. This may be downloaded from the Drivers and Utilities page on the FTDI Chip website. The so-called Virtual COM Port (VCP) drivers arrange for the interface to behave like an ordinary serial port. There are drivers for Windows, MacOS and Linux. Here, we will assume the Windows drivers are used.

Having installed the driver, the simulated COM port may be addressed by applications just like any regular serial port on the system. Higher programming languages like Delphi and C++ allow 'components' like Tcomport to be employed for the communication with the serial interface. If you do your own programming anyway, we'd recommend using the '**D2XX 'Direct' Drivers**' for Windows instead of the VCP ones. A Direct Driver **must** be employed to be able to program the external EEPROM!

The Windows VCP drivers copied form the FTDI website come in two flavours: with and without PNP. This has nothing to do with the interface proper (which is always recognised automatically by Windows). But only concerns the hardware connected to the interface (see the subheading 'Software').
The VCP driver download comes as a zip file containing drivers for Windows98, ME, 2000 and XP. The zip archive file is unpacked on the hard disk. The FTDI Chip website supplies extensive documentation, all the latest on the software and the installation process.

Windows will automatically launch its New Hardware Wizard when the interface is connected to the USB port on the PC. Next it will prompt you to point to the location where the drivers may be found. Browse and navigate to the folder containing the unpacked files. After a short delay, the Wizard will find FTDIBUS.INF and install the drivers and associated software for the interface.

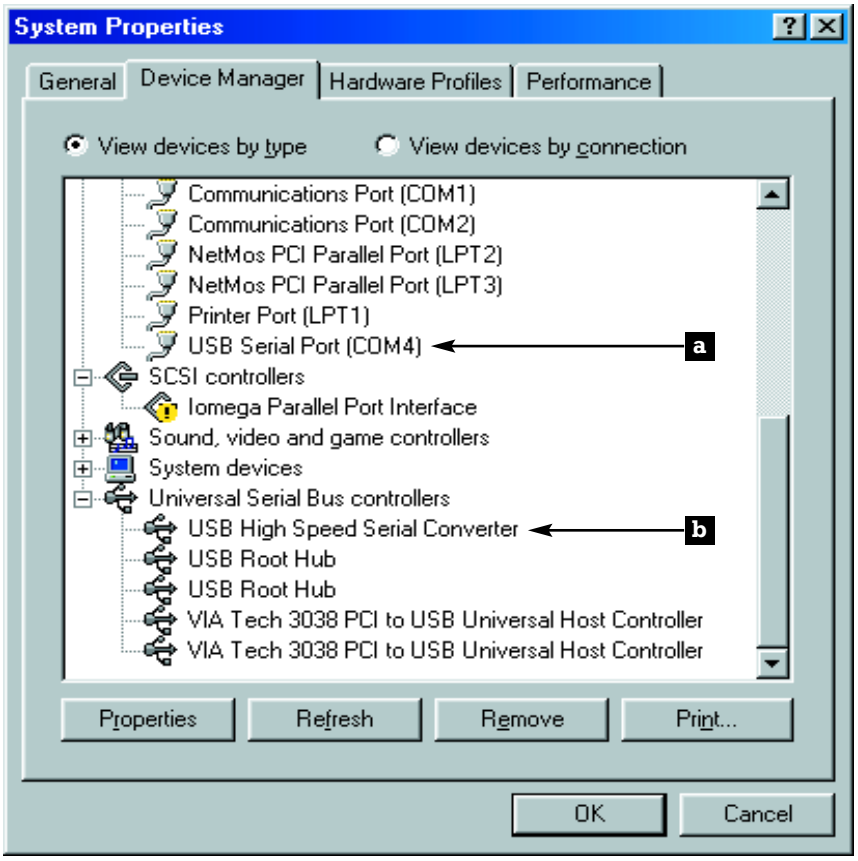


Figure 8. After the installation you should find a new 'USB Serial COM Port' under 'Ports' in the Device Manager (8a) and a new 'USB High Speed Serial Converter' under 'Universal Serial Bus Controller' (8b).

Note: *since writing this article we have been advised by FTDI Chip that the FT232AM chip has been superseded by the FT232BM.*

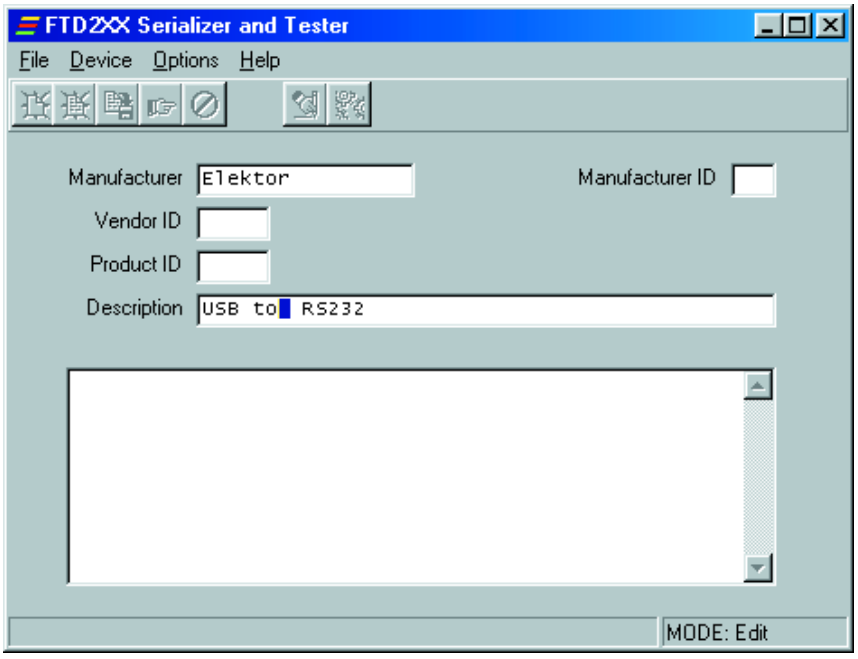


Figure 9. Utility program for (optional) programming of PID and VID information into the 93C46 EEPROM.

ensures that the USB side of the FT232AM appears as a USB device in the sub-folder 'Universal Serial Bus Controllers' (see **Figure 8b**).

Figure 9 shows the programming window of the tool available for programming the EEPROM connected to the FT232AM. This little tool may also be downloaded from the FTDI website. OEMs (original equipment manufacturers) may want to use this tool to program their own VID or PID. If you do not have a VID or PID, you may either omit them or omit the entire EEPROM. Yet another possibility is to simply resort to the PIDs and VIDs reserved by FTDI Chip. For the FT232AM, VID = 0403 and PID = 6001. A more extensive description of the actual uses of this option may be found in the programming instructions supplied by FTDI in the form of yet another free download.

Finally, we should mention that there are, of course, limits to the performance of a converter acting as a simulated RS232 port. Control of the data flow is essential to ensure trouble-free conversion between RS232 and USB, in order to prevent overruns occurring on the two buffers inside the FT232AM (128 and 384 bytes). Such situations would cause bytes to be lost in the conversion process and are most likely to occur at high data speeds when no handshaking is used.

(020375-1)

Literature:

– USB Driver Programming, Elektor Electronics October and November 2002.
– USB UART, Elektor Electronics December 2001 and January 2002.
– USB Interface, Elektor Electronics September 2000.
– SMD's? Don't Panic!, Elektor Electronics January and February 2003.

Free Downloads

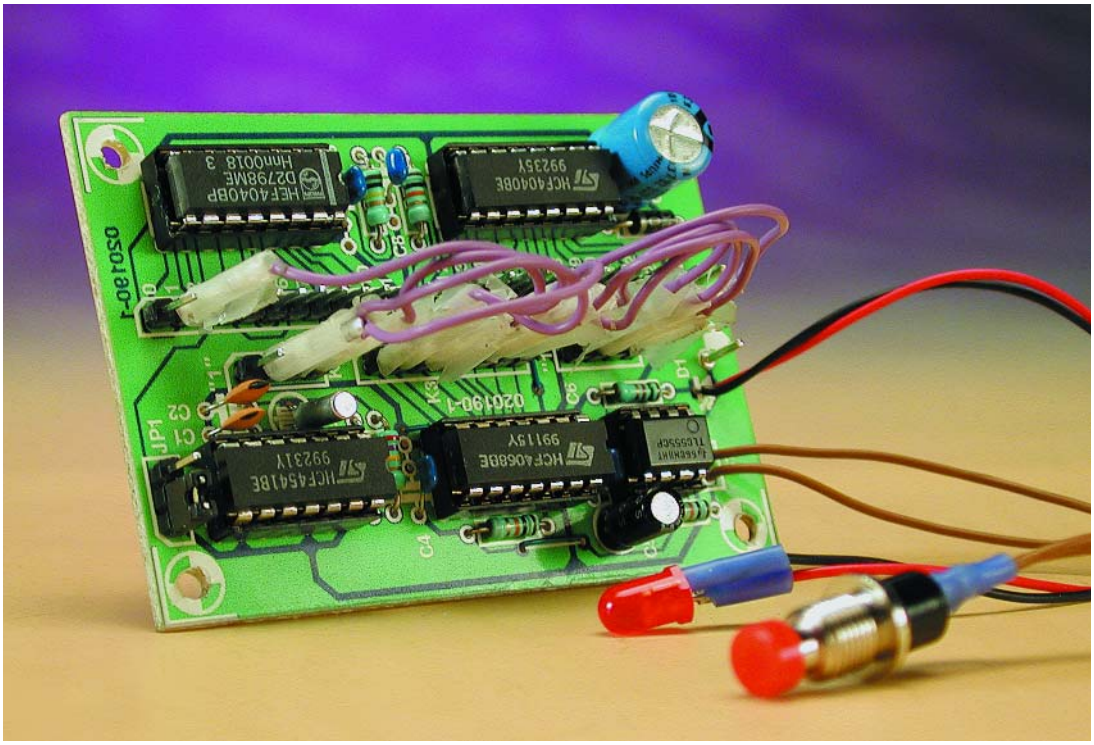
– PCB layout (pdf file), on www.elektor-electronics.co.uk/dl/dl.htm. Select file number **020375-11.zip**, month of publication.
– FT232AM datasheets, drivers, info on ready-made modules, etc., www.ftdichip.com

Week/Month Timer

Programmable for long periods

From an idea by Doug Pratt

As opposed to most timer circuits, this month's Mini Project is intended for very long periods. A switchable divider chain using two 4040 counter ICs allows periods of up to 194 days (yes that's 6 months) to be timed with great accuracy. Almost any other period shorter than the maximum may be set up, and all without a microcontroller or indeed any sort of exotic component!



It started out as a relatively simple project — a timer to remind us (and friends) to take our wheelie bins out ready for collection once a week (or month). The original intention was to make the timer drive a voice/record playback module programmed with a snappy message. However, for the purpose of this Mini Project, such an extension was deemed too complicated, so a rather simpler 'time-out' warning device was chosen, although it is still possible to hook up a voice module if you want to.

The vast majority of circuits capable of counting down predefined periods are intended for use as a cooking timer, darkroom timer or parking timer. Their timing range will typically cover a few minutes to a couple of hours. If you want to be reminded of your fortnightly wheelie bin chore, the 6-weekly appointment at the hairdressers, the 3-monthly check-up at the GP, or an equipment servicing interval, then you need to resort to a calendar or a diary because timers with such long periods simply do not exist. Oops, that should be "did not exist" because the timer described in this article is the exception to the rule. Here's why: it can be set to periods **within the range 2 to 16,777,216 seconds!**

Simple electronics

The circuit diagram (schematic) of the timer is given in **Figure 1**. As you can see, the project consists of just five plain integrated circuits and

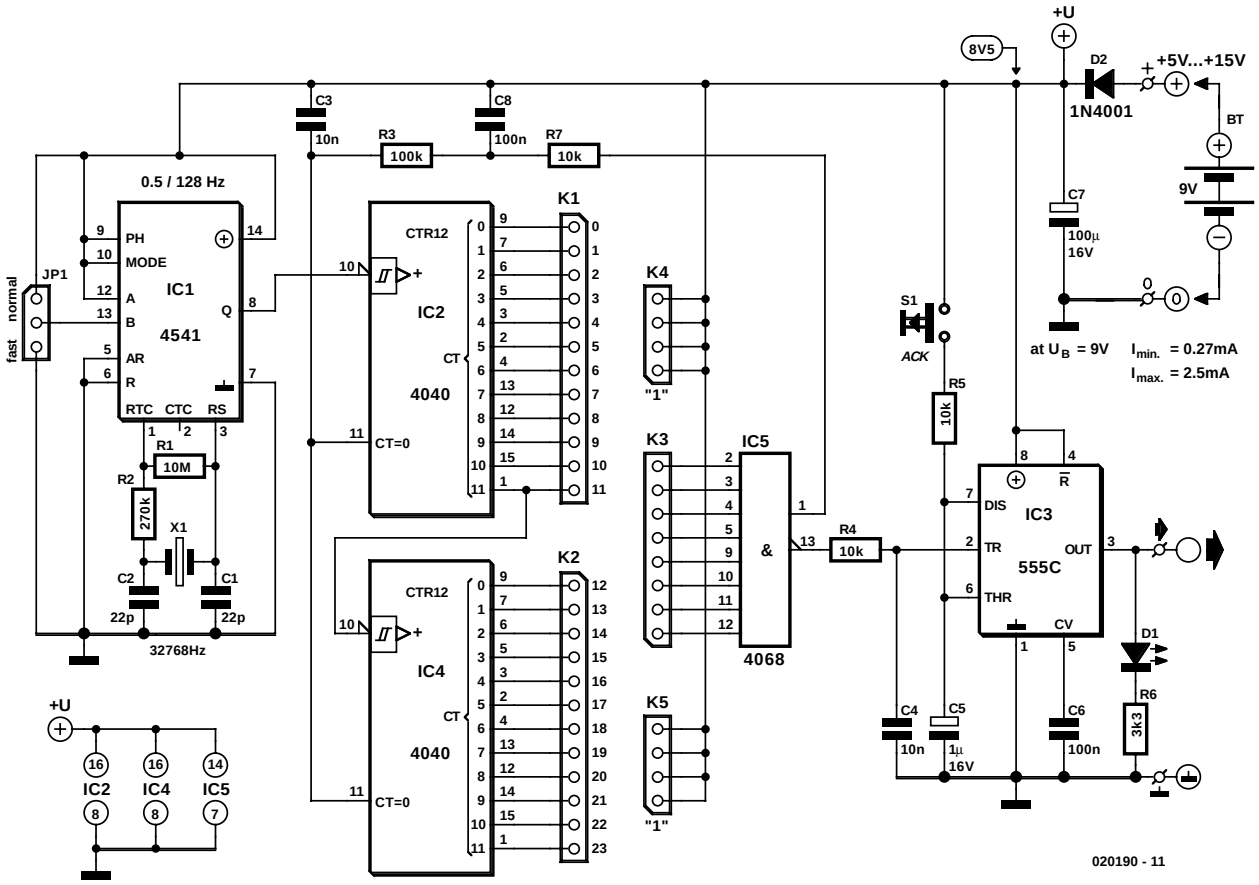


Figure 1. Cascaded binary counters enable the circuit to generate extremely long delays like weeks and months with 1-second accuracy.

a handful of discrete parts. Functionally, the circuit may be divided into three sections: an oscillator, a user-configurable counter and a flip-flop in control of an LED which acts as an indicator that the programmed period has elapsed. Note that the circuit allows for other visual or acoustic actuators to be controlled, including a buzzer.

IC1 and its surrounding passive components acts as the oscillator. The 4541 is a little known CMOS IC described by the manufacturers as a 'programmable timer'. In addition to an oscillator it contains a 16-stage binary counter which may be preset using pins A and B on the chip. Here, a divisor of 2^{16} has been chosen. Consequently, the quartz-generated clock frequency of 32,768 Hz is divided down to 0.5 Hz, which corresponds to a period of 2 seconds.

The 2-seconds pulse in turn clocks a divider chain ('cascade') consisting of two 12-stage binary counters type 4040 (IC2 and IC4). The ICs are cascaded by connecting the last output (Q11 on IC2) of the first IC to the clock input (CLK) of the second counter (IC4). In this configuration, the two 4040s allow a divisor of 2^{24} to be set, which results in the previously mentioned maximum time of 16,777,216 seconds. At this point some of you may wonder why the period is not twice as long — after all, the divider chain is fed with 0.5 Hz and not 1 Hz. The answer is simple: the output is activated as soon as **half** the period time of the counter has elapsed.

The divider cascade is followed by an 8-input NAND gate (IC5). By connecting one or more counter outputs (available on K1 and K2) to the input(s) of the NAND gate (available on K3), almost any desired period can be defined.

Once the preset value ('count') is reached, two things happen: to begin with, the Low-to-High transition that occurs at the non-inverting output of NAND IC5 is used to reset our two counters IC2 and IC4 by way of two R-C networks R7-C8 and

Table 1. Divisors.

Connection	seconds
K1-0 (Q0-IC2)	2
K1-1 (Q1-IC2)	4
K1-2 (Q2-IC2)	8
K1-3 (Q3-IC2)	16
K1-4 (Q4-IC2)	32
K1-5 (Q5-IC2)	64
K1-6 (Q6-IC2)	128
K1-7 (Q7-IC2)	256
K1-8 (Q8-IC2)	512
K1-9 (Q9-IC2)	1,024
K1-10 (Q10-IC2)	2,048
K1-11 (Q11-IC2)	4,096
K2-12 (Q0-IC4)	8,192
K2-13 (Q1-IC4)	16,384
K2-14 (Q2-IC4)	32,768
K2-15 (Q3-IC4)	65,536
K2-16 (Q4-IC4)	131,072
K2-17 (Q5-IC4)	262,144
K2-18 (Q6-IC4)	524,288
K2-19 (Q7-IC4)	1,048,576
K2-20 (Q8-IC4)	2,097,152
K2-21 (Q9-IC4)	4,194,304
K2-22 (Q10-IC4)	8,388,608
K2-23 (Q11-IC4)	16,777,216

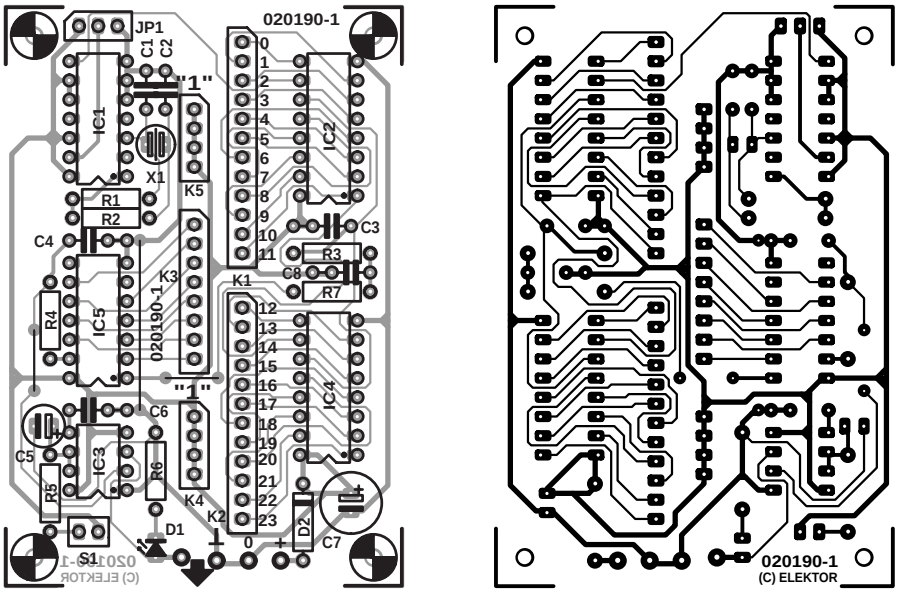


Figure 2. The PCB designed for the Week/Month Timer is small and together with the 9-V PP3 battery will easily fit in a standard plastic case.

COMPONENTS LIST

Resistors:
R1 = 10MΩ
R2 = 270kΩ
R3 = 100kΩ
R4,R5,R7 = 10kΩ
R6 = 3kΩ3

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

Semiconductors:
D1 = LED, high-efficiency, 2 mA
D2 = 1N4001
IC1 = 4541
IC2,IC4 = 4040
IC5 = 4068
IC6 = 555C (CMOS version)

Miscellaneous:
JP1 = jumper
K1,K2 = 12-way SIL pinheader
K3 = 8-way SIL pinheader
K4,K5 = 4-way SIL pinheader
S1 = pushbutton, 1 make contact
X1 = 32.768kHz quartz crystal
9-V PP3 (6R22) battery with clip-on wires
PCB, available from The PCBShop (see Elektor website)

R3-C3. This causes the user-defined timing period to be started over again. Secondly, the High-to-Low transition at the inverting output of IC5 is fed to the flip-flop in order to trigger it.

IC5, a 555 timer in so-called bistable mode acts as a flip-flop. The flip-flop will toggle, pulling its output High, on the falling edge of the pulse applied to the TR– input. Its pevious state is restored when a positive pulse transition (rising edge) at the TR+ input. So, when the preset period has elapsed on the timer, the output of IC3 goes High causing LED D1 to light. If you are not satisfied with just the LED, the same output may be used to drive an active buzzer or another low-power actuator. The user can acknowledge the 'time-out' warning (whether visual or acoustic) by pressing pushbutton S1. This causes a positive pulse to be applied to the TR– input and the flip-flop to return to its non-active state.

Some more details

A couple of details in the circuit have escaped discussion so far.

As indicated in the circuit diagram, jumper JP1 provides a selection between 'fast' and 'normal'. This extra feature has been added for test purposes. When 'fast' is selected, pin B of IC1 is pulled to ground. This causes the divisor in IC1 to be changed such that the output frequency at the Q pin is increased by a factor of 128. This is useful when testing the circuit because you don't have to wait so long to see you got the 'programming' right (more about this further on).

Because the counters work a-synchronously (they all operate in 'ripple' mode i.e., consecutively), glitches may occur at the output of NAND gate IC5 and these may cause false timing results. The problem is solved by R4-C4 which suppress glitches and spikes. The propagation delay of each counter in the 4040 ICs is slightly dependent on the actual supply voltage and will vary between 50 ns and 100 ns per counter stage. For IC2 and IC4 together, this works out at a maximum of (about) 24 × 100 ns = 2.4 μs.

Of course, the non-inverting output of the NAND gate may also produce glitches which when not suppressed could easily cause false resetting of the counters. Here, too, an R-C network is in order, this however has a potential problem because it causes the reset voltage to increase slowly. Assuming that the two counters are not exactly identical (which unfortunately is always the case) then one of them will be reset while the reset threshold for the other is not yet reached. That's annoying because the resetting of one of the counters will cause the reset pulse to disappear, resulting in just one of the counters being cleared to zero and the other still at its previous count. To prevent this from happening, a double R-C network R7-C8, R3-C3 is applied which gives the reset signal a quite different shape: there now is a 'dead' time, which is another way of saying that the entire pulse is slightly delayed. The upshot is that it not only takes a little longer for the pulse to reach the reset level, but also that the pulse is allowed to 'rise' further by about 0.5 V, to make absolutely sure the other counter is also reset.

How to set the period

Calculating which outputs you have to use to set up a certain period is quite simple, the process being similar to doing a long division.

Step 1. Calculate the desired time in seconds. Let's take one week as an example: 60 seconds × 60 minutes × 24 hours × 7 days = 604,800 seconds.

Step 2. Use the divisor table (**Table**

1) to look up the first number that's smaller than the above value. You'll find Q4 on IC4 with the associated divisor 524,288. Now determine the difference:
604,800 – 524,288 = 80,512

Step 3. Repeat step 2 until the remainder is zero:
80,512 – 65,536 = 14,976
(output Q1 on IC4)
14,976 – 8,192 = 6,784
(output Q0 on IC4)
6,784 – 4,096 = 2,688
(output Q11 on IC2)
2,688 – 2,048 = 640
(output Q10 on IC2)
640 – 512 = 128 (output Q8 on IC2)
128 – 128 = 0 (output Q6 on IC2)

Each of the counter outputs found in this way has to be connected to a NAND gate input. Any non-used inputs on the gate have to be tied to the positive supply rail, hence the presence of connector strips K4 and K5.

Construction

The artwork designed for the printed circuit board is shown in **Figure 2**. The ready-made board is not available from Readers Services but may still be obtained through an alternative channel called **The PCBShop** (EuroCircuits). Alternatively you may want to etch your own board using the artwork printed here (or downloaded as a .pdf file from our website). The construction itself should not cause problems even to relative newcomers.

If you intend to use the counter for just one, fixed, period, then the necessary links between K1/K2 and K3 may be simple wires permanently installed on the circuit board. The same goes for the unused inputs on the NAND gate.

On the other hand, if you (like us) envisage having to use different long periods, it is better to fit SIL pinheaders in positions K1-K5. Selecting the desired period is then easily done using a set of short cables with a pin connector at each side. The pin connectors should be isolated using heat shrink sleeving or similar.

Power supply

The Week/Month Timer described here is very economical in use, drawing just 0.3 mA with the LED off. When the LED comes on, the current drain rises to about 2 mA. Obviously, low power consumption is a must because in practice the circuit will be powered from a battery. Today's PP3 9-V alkaline batteries have a nominal capacity of about 500 mAh which is sufficient for about two months of continuous operation of the timer (provided of course you do not leave the LED on for weeks on end). If this sort of battery capacity is insufficient for your application, consider using a pack of four penlight (AA) batteries in a holder, or a mains adapter with a stabilized output voltage between 5 and 15 volts dc.

(020190-1)

Temperature Indicator for the PC

Useful and decorative!

Design by A. Köhler

Alongside its purely functional uses the PC also has a position as a status symbol. At computer nerd get-togethers, it is *de rigeur* to have some novelty to show off. And if it also serves some useful purpose, so much the better.

Enclosures with windows and internal illumination have already started to become available, showing off the innards of a PC. There is even a special word — *modding* — for this type of PC (re)construction. As well as various modifications whose benefits can be argued over at considerable length, there are those which have a definite useful purpose.

For example, the temperature inside a PC's enclosure is an important quantity. Depending on the processor, the fans and the ambient conditions, the temperature can rise to the point where trouble-free operation can no longer be guaranteed. There are special displays available, often in conjunction with fan systems, that offer a temperature readout. Unfortunately these displays occupy a full 3.5 inch or 5.25 inch drive bay in the PC. But remember that PCs have a red, yellow or green LED that serves as a power indicator. Why not use this simple indicator to offer extra functions, such as showing the temperature inside the enclosure?

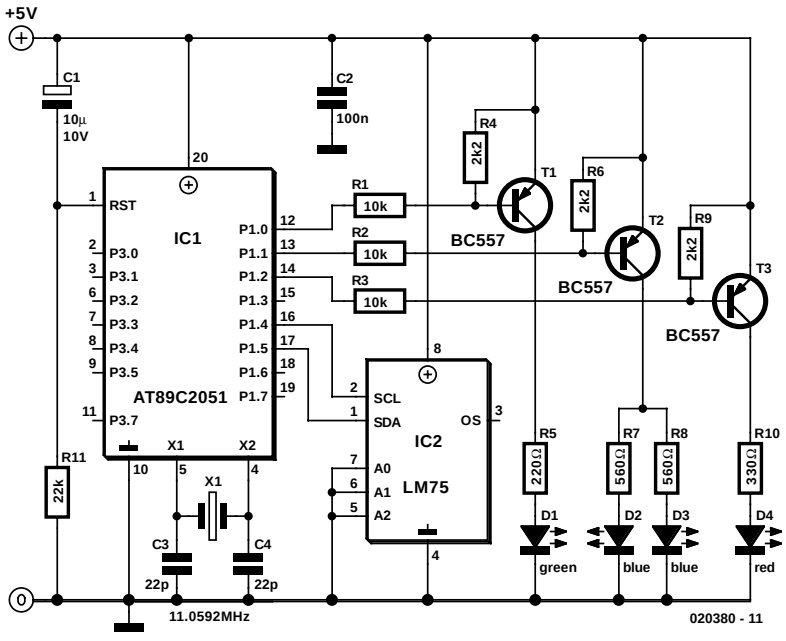


Figure 1. The simple circuit diagram of the temperature indicator.

Before we start

Before we can build a temperature indicator we must determine the range of temperatures over which it is to operate. Intel and AMD specify their processors for a temperature of 40 °C to 45 °C inside the case. Without expensive specialised cooling it is not really possible to effectively remove the heat generated by the processor at case temperatures above 70 °C. The maximum junction temperature of 125 °C is in any case the ultimate limit. An office PC will usually be used in a more-or-less air-conditioned environment (the *Elektor Electronics* offices being a particularly notable exception). And hardly any user will want to operate their PC at temperatures below 15 °C, so the PC's components are rarely below this temperature.

It is therefore not at all unreasonable to set the measurement range to +10 °C to +70 °C. Using these values we can use the top surfaces of hard disks and heatsinks as measurement points. The temperature sensor used should be a readily available and inexpensive type, such as the LM75 (or DS75). Reading the temperature out can be done using a

simple microcontroller such as the AT89C2051, which will allow us to transfer the temperature value over an I²C bus. We can use a three-colour LED as an indicator, replacing the one-colour power indicator LED usually provided. The multicolour LED is constructed from four LED chips in three colours: one red, one green, and, because of their lower light output, two blue chips. In order to give the impression of a single composite colour, a diffused package is used. If the individual LED chips can still be made out, adding an extra sheet of diffuser will result in the desired effect. The impression of different colours is achieved by driving the individual LED chips using pulse-width modulated signals from the microcontroller.

The circuit

Figure 1 shows the complete circuit, using an 89C2051 microcontroller from Atmel. This device's baby brother, the 89C1051 (which offers 1 kbyte of memory) can also be used. The reset circuit on pin 1 is not critical: the capacitor value can be anything from 1 µF to 10 µF, the resistor anything from 10 kΩ to 50 kΩ. Likewise, the oscillator circuit between pins 4 and 5 is not critical: crystals from 7.3728 MHz to 14.318 MHz have been tested. The three-colour LED is connected via a PNP transistor driver. A 5 mm LED in a diffused package is used: suitable devices are available from, for example, Reichelt Electronics (www.reichelt.de) and Conrad Electronics (order code 185353, www.int.conradcom.de). The LED is manufactured by Kingbright under part code LF59EMBGMBW. Note that many devices advertised as 'three-colour LEDs' are not suitable for this application. The Kingbright device you need has six terminals, see also the datasheet which may be found on the Conrad webserver.

Each BC557 transistor has a series base resistor and a pull-up resistor to ensure that the transistor is turned fully off. Because of the different forward voltages and different light intensities of the LEDs, different values of current-limiting resistor are required. These resistor values can be adjusted, since the LED has reasonably wide tolerances.

Port 1.0 controls the green LED, port 1.1 the blue LED and port 1.2 the red LED. The LED current should be around 20 mA. Communication with the temperature sensor is over ports 1.4 (the SCL clock signal) and 1.5 (the SDA data signal). The LM75CIM5 temperature sensor (the 5 V version) has its write address set to 90H and its read address set to 91H by the levels on A2 to A0.

Software

The most interesting part of this project is the software. The software is available as source and compiled on the Internet from the *Elektor Electronics* website (which is free) and on diskette **020380-11** (for which there is a nominal charge). If you cannot, or do not wish to, program the device yourself, a ready-programmed microcontroller is also available (order number **020380-41**).

The details of the program can be seen from the listing. The program uses pulse-width modulation, controlled by a program loop rather than using the internal timers. The program begins by initialising the port outputs. A register is allocated to each colour to store the time for which the corresponding LED is illuminated. Next the program selects the temperature register of the LM75 and reads out the temperature value stored there. Only the four most significant bits are used, the four least significant bits of the temperature reading being ignored. Since the full temperature range of the device is not used, the temperature scale is shifted to start at 10 °C by subtracting 0AH from the value read. This value is to be converted into a 'colour value'. Each colour value consists of three time constants which determine for how long the corresponding primary colour LED is illuminated. The time constants are obtained from a table indexed by the temperature. Routine ZAU causes the colour value to be displayed.

The data pointer is used to point into the table containing the illumination time values. Each temperature value uses three bytes of table space. The first byte gives the time for which the green LED is illuminated. The green LED is turned on and a delay loop is entered. The same happens for the blue and red LEDs. The primary colour LEDs are driven inside a loop which is executed 255 times, so that the eye cannot distinguish the individual colours. Instead, the impression of a particular colour and intensity is given, depending on the time constants and hence on the temperature.

A particular point to note is that a value of 00H for a time constant corresponds to the maximum possible illumination period, since the first time the DJNZ instruction is executed in the loop an overflow (or rather, a borrow) occurs. With a value of 01H the illumination period is so brief that it is practically invisible to the eye. Delay routine ZEIT is used to extend the (otherwise very short) illumination period. If the crystal frequency is changed, it may be necessary to adjust this delay somewhat. The values given have been tested with a 11.0592 MHz crystal. The colours themselves can be changed by altering the values in the table.

The example program given indicates low temperatures by a blue colour. As the temperature rises, so does the proportion of green in the colour. At 35 °C the green LED alone is lit. If the temperature climbs further, red is added into the colour. Over 50 °C only the red LED lights.

Since the maximum offset available is 255 (an 8 bit register is used for the temperature), the table can only be extended to about 70 °C, but above this temperature an office PC is unlikely to operate correctly in any case. Communication with the temperature sensor uses the I²C bus routines START, STOP, READ and WRITE in their minimal configuration. Error handling is not provided as transmission errors on the I²C bus will not have serious consequences. With the source listing to hand there should be no difficulty in making modifications to the display.

(020380-1)

Free Download

source code file (in German) and hex file for programming the microcontroller. File number: 020380-11.zip.
www.elektor-electronics.co.uk/dl/dl.htm
(select month of publication).

Portable MP3 Players

Smaller and with new ICs

By Harry Baggen

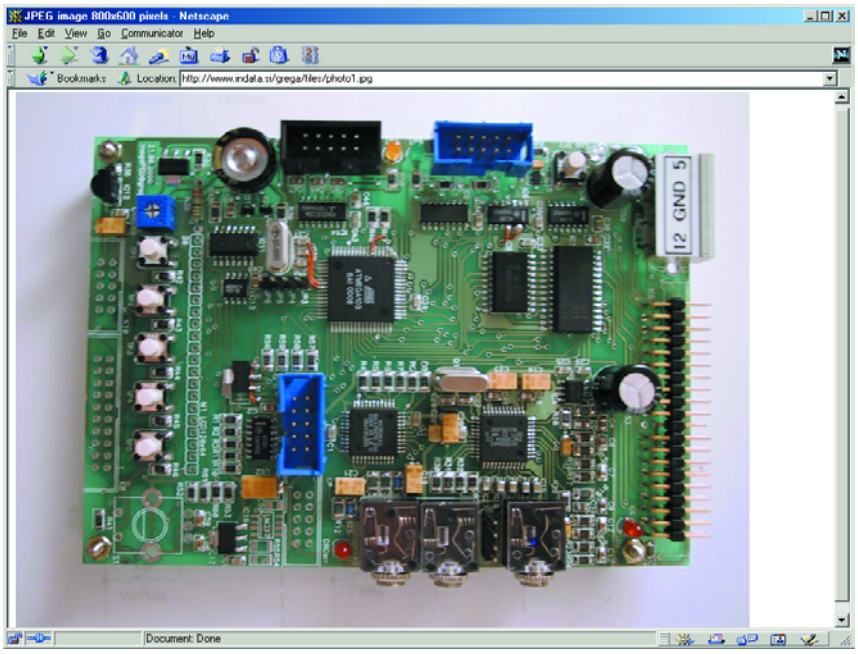
The MP3 format took just a few years to become an important standard in the music industry. Today, MP3 players come in many brands and models, both stand-alone and portable, with either a CD drive or a Flash memory to hold music files. Despite the wide choice of commercial units now in the shops, building an MP3 player yourself is still great fun.

It is fair to say that the MP3 format triggered a revolution in the musical industry. Thanks to MP3's enormous compression factor it became possible, for the first time, to distribute music files using the Internet and even email. But MP3 also has distinct advantages for use at home and 'on the road'. After all, a single CD-ROM has sufficient capacity to hold hours of music, while MP3 players with an internal hard disk can easily store a complete musical collection.

Admittedly this is the third time the Electronics Online column covers DIY MP3 players, but this seems justified because of the great interest in the subject. Thanks to the arrival of new MP3 decoder ICs like the VS1001 from the Finnish manufacturer **VLSI Solution** [1], designing your own MP3 player has become more attractive than ever before.

Looking at a number of projects it is clear that designers of MP3 players increasingly opt for the 'stand-alone variant' equipped with a Compact Flash memory or a 2.5-inch hard disk. Both file storage media guarantee a compact unit which, in some cases, can be called 'portable'. Indeed, some of the models referred to on these pages are so small and smartly styled that they are at least potential competitors of commercial products.

The **Super Simple pocket size MP3 player** [2] is a design based around a PIC 16LF877 from Microchip. At the heart of the circuit we found the above mentioned VS1001 chip from VLSI Solution. The memory medium is a CompactFlash card. The whole player is smaller than a cigarette box and represents a kind of bare bones design. The designer has added no



'luxury' whatsoever, even a display is conspicuously missing. None the less, the circuit is perfectly capable of playing back files stored in the Flash card while the PIC should have enough throughput to add bells and whistles.

YAMPP (yet another MP3 player) is an MP3 DIY website that's been around for a few years already. Having published his seventh design, the website author can be called industrious by all standards. The **YAMPP-7** [3] is a nice, small and portable MP3 player employing an ATMega161 controller and a VS1001 decoder IC. The design also comprises a USB interface and a Li-Ion battery charger. Just as the battery, the display has been salvaged from a GSM phone The designer offers individual components as well as ready-populated boards for this project.

Since they were first mentioned in this column (June 2001), the technical boffins behind the **MP3ar** website [4] have been very active and now present an MP3 portable sporting a 2.5-inch hard disk. The controller applied is a PIC 18C452 while the decoder chip is once again a

VS1001. A complete kit consisting of a ready-populated board, an LCD and a CD-ROM wit software can be ordered for about $100. The only item you have to add yourself is a hard disk drive.

Although Gregor Horvat's **megaPEG player** [5] design has not seen any updates for about a year, the pictures on the website show a nice design of a compact board that allows a graphic display and a hard disk to be hooked up. In his design Gregor employs an ATMega103 controller and the good old MAS3507D decoder chip.

Fallguy [6] is a universal design that can be built and extended to personal requirements for use at home, in the car, or as a portable player. The hardware consists of, among others, a 68HC912 and an STA013. An LCD with 128x64 pixels shows the track titles and operating functions. A special feature of this player is its ability to connect to two hard disk drives (master/slave) allowing access to a whopping 256 Gbytes of memory**!**

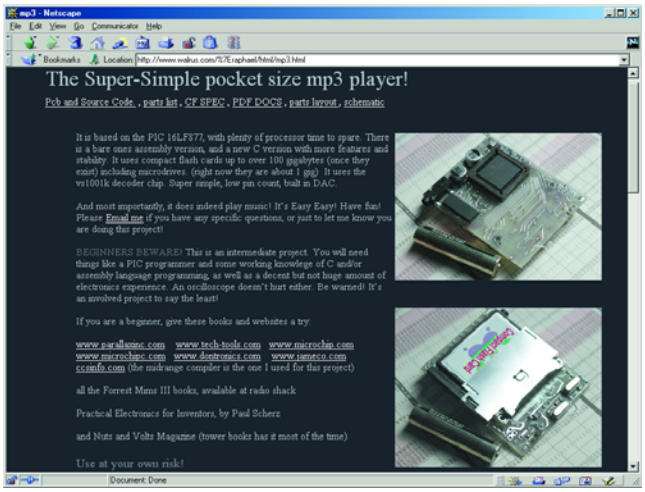
The Beatbox 2002 [7] is by no means a miniature-size design, but still a compact MP3 player. The size of this player is about equal to that of a 3.5-inch hard disk drive. It consists of a main board, a 2.5-inch hard disk and a large graphic display with touch-screen operation. A Motorola MC68332 processor looks after all control functions, assisted by a VS1001 decoder chip. A USB 2.0 interface ensures fast data transferring of MP3 files into the player. This player offers many functions and is compatible with various audio file formats like MPEG-1, MPEG-2 layers 1, 2 and 3, and MPEG 2.5.

The **IMAP2** player designed by David Freudenberger [8] also falls in the category 'compact but not exactly portable' and may be used with a CD drive or a hard disk. Here, too, a PIC is used for the central control functions, while the familiar MAS3507D from Micronas looks after the processing of the MP3 information.

A similar design (PCB with hard disk and display in a compact case) has been dubbed **MP3 Player** [9] by Codepuppies. Because the most recent update dates back one and a half year, constructors are well advised to check component availability before embarking on the project. The central parts in the design are a PIC and a MAS3507D.

The website **mp3projects.com** [10] was mentioned two years ago already, bit still ranks as the best source if you need an overview of all important MP3 DIY projects currently on the web. The projects overview comprises several pages and is subdivided into three main groups: stand-alone, portable and PC-hosted. For $20 the authors of the website also offer a CD-ROM containing a variety of software related to home construction of MP3 players. The offerings include datasheets, assemblers/disassemblers and PCB layout programs, as well as source code files, snippets and routines for all manner of tasks in an MP3 player.

(035026-1)



Internet adsresses

[1] VLSI Solution: www.vlsi.fi/
[2] Super-Simple pocket size MP3 player: www.walrus.com/%7Eraphael/html/mp3.html
[3] YAMPP-7: www.myplace.nu/mp3/index2.htm
[4] MP3ar: www.mp3ar.com/
[5] megaPEG: www.indata.si/grega/megapeg.htm
[6] Fallguy: www.loetronic.com/home-e.htm
[7] Beatbox 2002: www.beatbox2002.de/
[8] IMAP2: www.chez.com/imap3/index2_e.html
[9] MP3 Player: www.codepuppies.com/~ben/sens/pic/mp3/
[10] mp3projects.com: www.mp3projects.com/