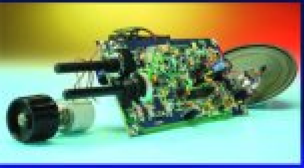
MARCH 2002
£3.20

ELEKTOR ELECTRONICS

THE ELECTRONICS & COMPUTER MAGAZINE

www.elektor-electronics.co.uk

AVR Programmer



VHF Airband Receiver

Light Barrier for Speed Measurement

how fast is your model car?

Relays on DMX

EEDTS-Pro v.1.2

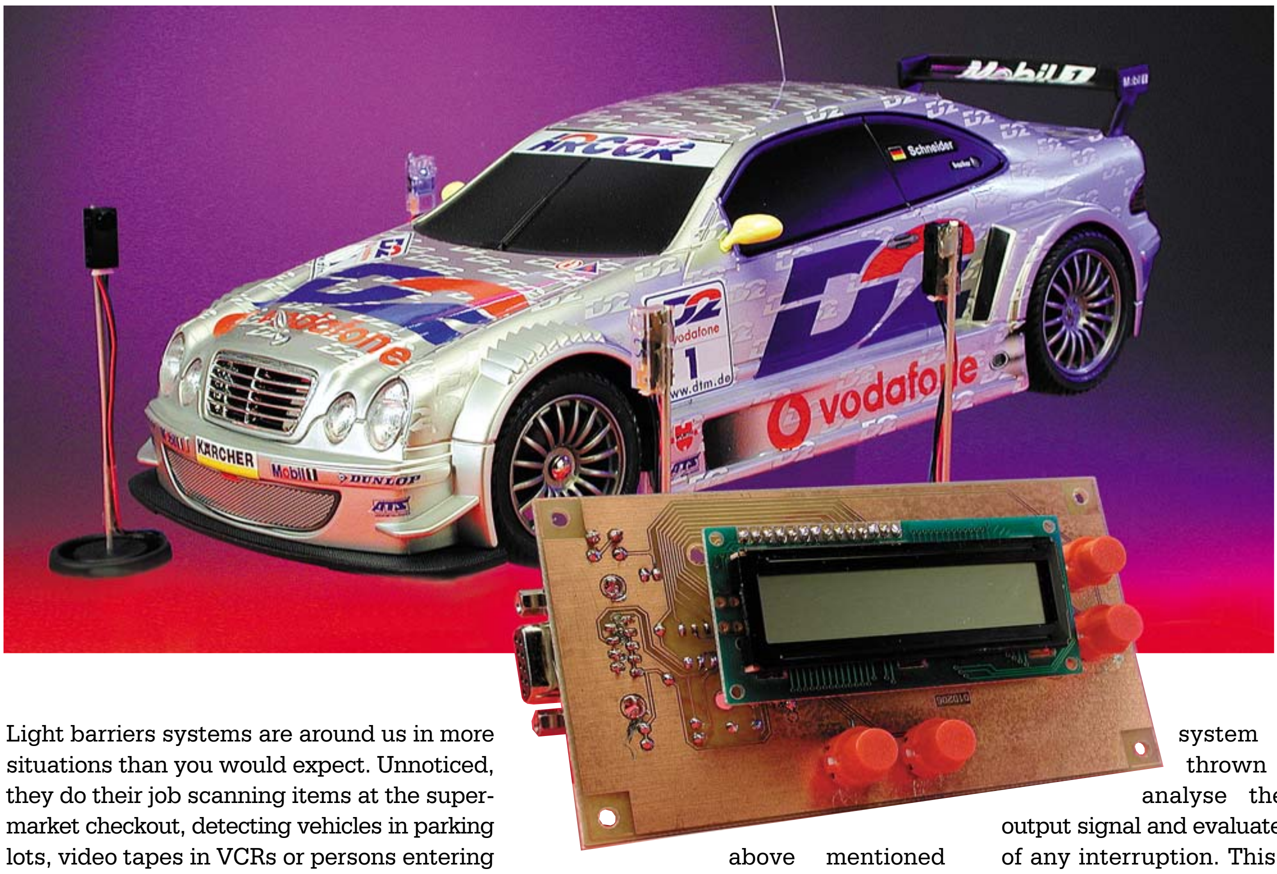
DTMF Codelock

Audio Frequency Notch Design

Speed Measurement System

A DIY Gatsometer with infra-red light barriers

The light barrier system described in this article allows accurate measurement of the absolute speed of model cars, planes and other moving objects. In education and training programmes, the system forms a perfect contactless speedometer. Over to you to football and golf fans to see who has the meanest ball kick or club swing.



Light barriers systems are around us in more situations than you would expect. Unnoticed, they do their job scanning items at the supermarket checkout, detecting vehicles in parking lots, video tapes in VCRs or persons entering or leaving buildings. Usually, infra-red light is employed because it is invisible to the human eye and ensures a high degree of immunity against interfering light sources.

Whereas simple light barriers operating without modulation, like the slotted light barriers that check if a CD or disk is inserted in the PC, have relatively simple functions, the above mentioned applications in parking lots, garages and security systems have far more challenging requirements in respect of 'noise' immunity. The solution, in nearly all cases, is to employ a 36-kHz carrier. The receivers are then designed to respond only to changes in the frequency. A microcontroller system is then thrown in to analyse the receiver output signal and evaluate the length of any interruption. This is done to prevent erroneous triggering of an alarm in the case of, say, an insect flying or crawling through the barrier.

As illustrated in Figure 1, speed measurement using a light barrier may be achieved with just one sender and one receiver. A disadvantage of this system is the risk of inconsistent measurement results if the moving object has an irregular shape causing multiple interruptions of the light beam. In addition, there's no assurance that the measured object follows the same, straight path when travelling through the barrier, again causing unreliable results.

Several infra-red detector ICs have been used in many projects published in this magazine over the past few years. One of these, the SFH505 from Siemens, has become a kind of standard component for this function. The SFH505 and a other functionally compatible devices (including IS1U detectors from Sharp) are found in lots of equipment of the 'consumer electronics' variety. Although these components are versatile and cost effective, they are less suitable for the project we have in mind, mainly because of the following aspects.

– The output response time to beam interruption is subject to too many time tolerance factors;
– For an acceptable range, the infrared light may not be modulated all the time. Pulses need to be inserted to prevent the control electronics from considerably reducing the receiver sensitivity.
– Because of the required pauses within the modulated signal, reproducible speed measurement is not feasible mainly because beam interruption (by the object) can not be distinguished from a pause (inserted by the system).

Despite the extra hardware, the system proposed in Figure 2 is the better alternative, if only for its much higher accuracy and more reliable measurement results when used to measure the speed of objects passing through the barrier.

To make real-life application of the above system as flexible as possible, the design allows the user to set up the length of the path the object has to travel along in order to interrupt the two light beams. In practice, this is done by placing the first and second barrier at a suitable distance from each other, and 'informing' the system about the exact distance before the measurement commences.

Features

– Speed measurement within range of 0.01 to 999 km/h
– Speed readout in m/s or km/h on 1x16 character dot matrix display
– Elapsed time readout (max. 16.777215 seconds)
– Resolution 1 µs
– Light barrier distance adjustable between 1 and 255 cm
– Optical alignment aid
– Single or continuous measurements
– Powered by 9-V battery
– Current consumption approx. 45 mA.

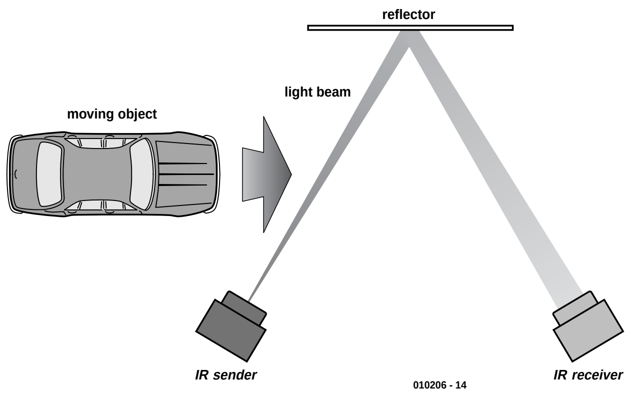


Figure 1. Speed measurement using one light barrier and a reflecting device.

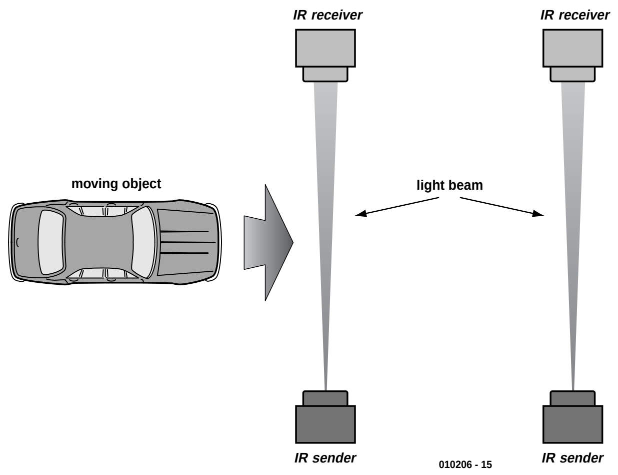


Figure 2. Speed measurement using two light barriers.

Sender and receiver

The Kodenshi Company from Korea have developed two modules (Emitter PIE-310 and Detector PID-310) for light barrier applications. Thanks to their high degree of integration, these modules contain all components necessary for a long-range yet noise immune light barrier system.

From the datasheets (in Korean) we were able to distil the following important characteristics:

– IR LED with internal modulation and associated photo detector in a separate, compact case with a lens system.
– Size approx. 17 × 8 × 7 mm
– Range 1 to 8.5 m (3 to 25 ft.)
– Active-low open-collector output
– Control input on sender
– Highly tolerant of ambient light thanks to optical filters and internal modulation.
– Low-cost sensor applications at large distances
– Suitable for use in Reflection mode
– 3-pin wire connection
– Supply requirements 5 V/5 mA (receiver) and 5 V/15 mA (transmitter)
– Switching speed 0.5 ms
– Half-sensitivity angle ±5 degrees
– operating temperature –10 to +60 °C

The modules, whose pinouts are given in Figure 3, are intended for use in paper sen-

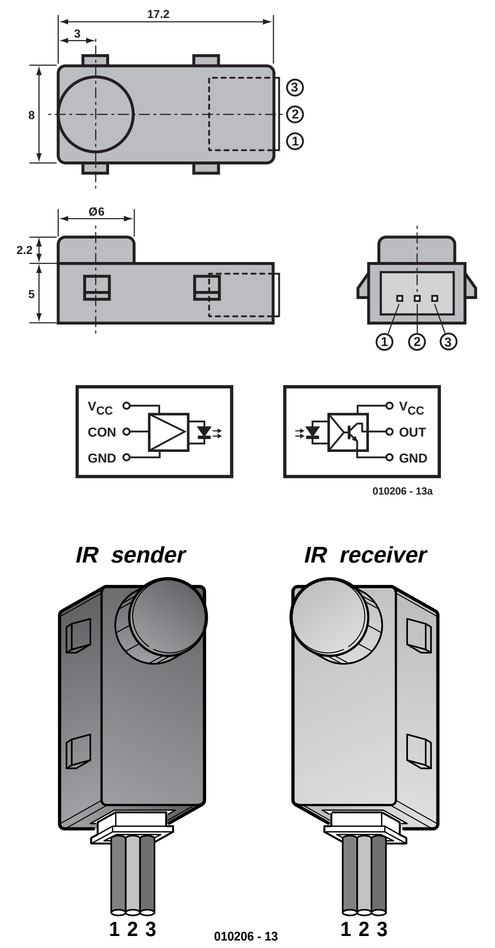


Figure 3. Pinout of the sender/receiver combination.

sors, distance sensors in reflection mode, counter and registration systems or proximity detectors.

The simplicity, relatively low cost and reliability of the circuit to be discussed is mainly due to these ready-made modules from Kodenshi. The module connection cables are available as accessories. They are essential, however, because it is very difficult to solder wires to the device pins.

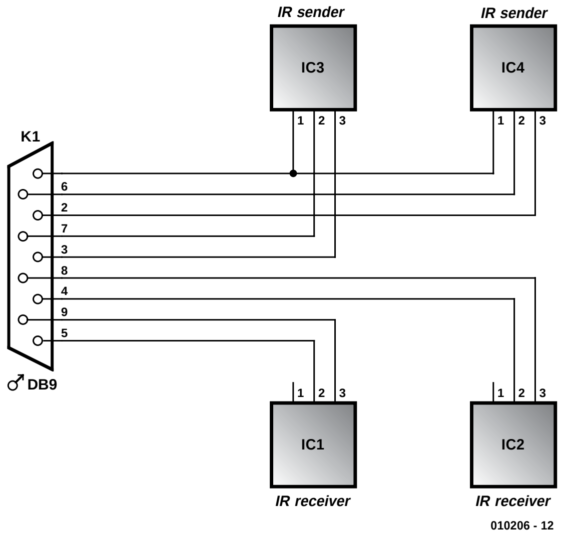
The microcontroller

By virtue of a programmed microcontroller, the circuit is relatively uncluttered. The circuit shown in Figures 3 and 4 consists of two light barriers, a display for the speed readout and system settings, four pushbuttons and, of course, the central microcontroller.

A ‘low power, low price, low pin count microcontroller’ (to use the words of Philips) type 87LPC762 is employed here. This device is based on the well-established 8051 architecture. Here, it is clocked at 6 MHz (1 µs cycle time), in view of the calculation load and the high time resolution required by the system.

The main features of the 87LPC762 may be found in the datasheet:

– 2 KBytes ROM
– 128 Bytes RAM
– 32 Bytes user programmable EEPROM
– 2.7 to 6 V supply voltage
– Two 16 Bit Timer/Counter
– Integrated Reset
– Internal RC oscillator for optional use
– 20mA drive capacity at all port pins



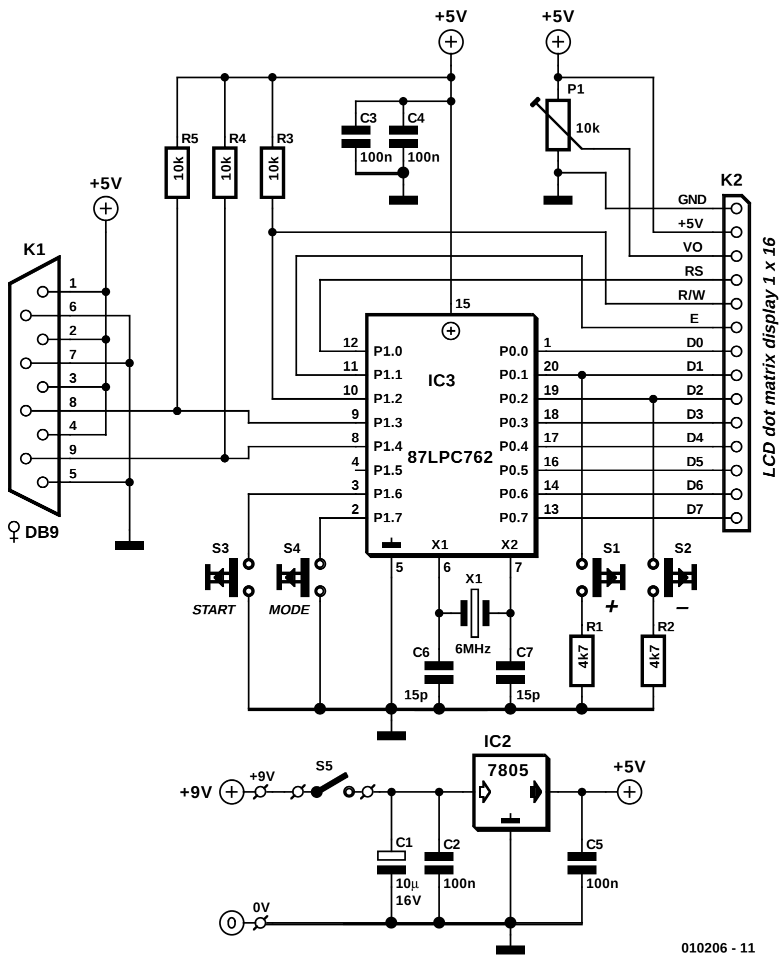
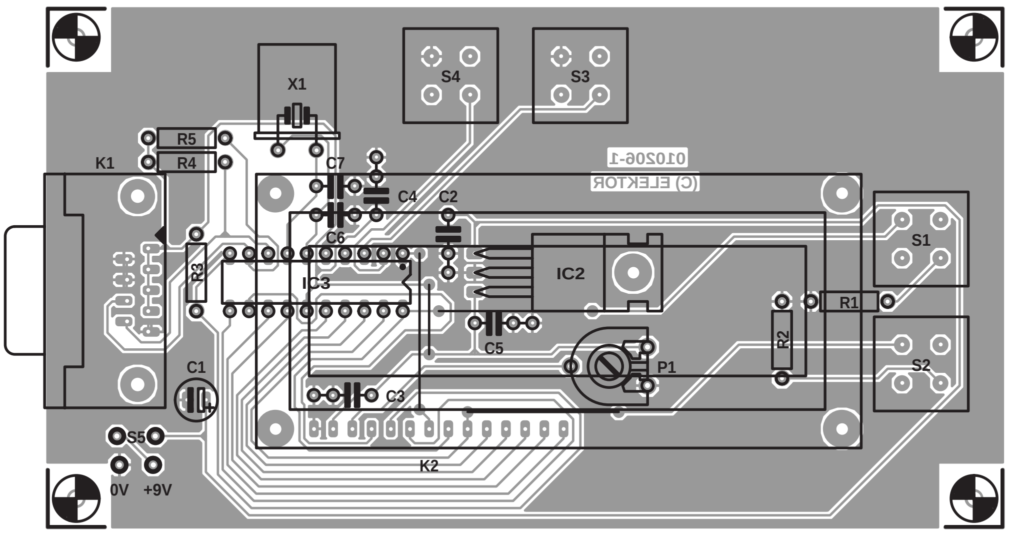


Figure 4. Circuit diagram of the sender/receiver combination.



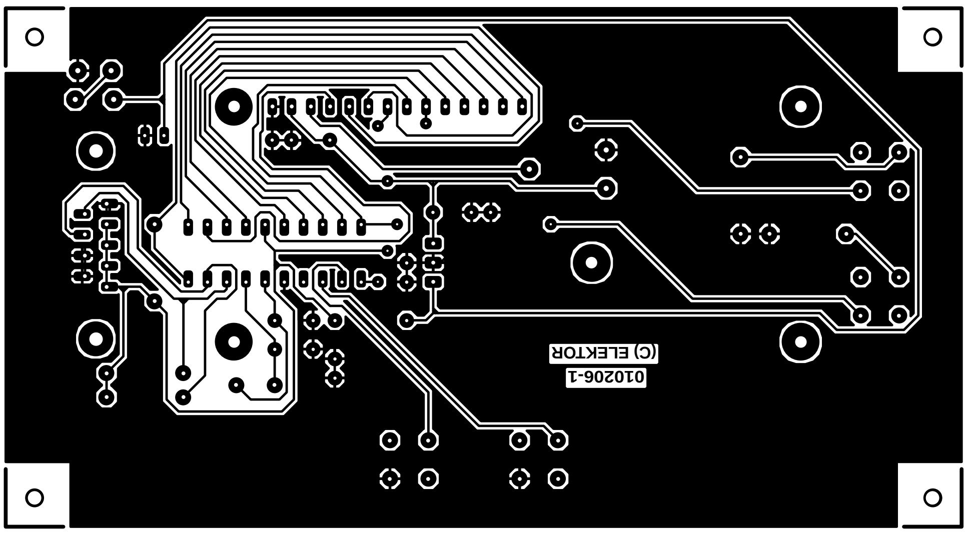


Figure 5. PCB layout for controller and display (board available ready-made).

COMPONENTS LIST

Resistors:
R1,R2 = 4kΩ7
R3 = 10kΩ
R4,R5 = 100kΩ
P1 = 10kΩ preset

Capacitors:
C1 = 10μF 16V
C2-C5 = 100nF
C6,C7 = 15pF

Semiconductors:
IC2 = 7805
IC3 = 87LPC762, programmed, Publishers' order code **010206-41**

Miscellaneous:
K1 = 9-way sub-D socket (female), angled pins, PCB mount
K2 = 14-way SIL pinheader
S1-S4 = pushbutton
S5 = on/off switch
X1 = 6MHz quartz crystal
LCD dot matrix display, 1x16 characters, 44780-compatible with connections in top left hand corner
2 combinations of IR-Sender PIE-310 and IR receiver PID-310 (Kodenshi, Farnell #139-865)
4 module connecting cables, Farnell #310-0728
PCB, Publishers' order code **010206-1**
Disk, contains source code and Hex files, Publishers' order code **010206-11** or free download from www.elektor-electronics.co.uk

– up to 18 I/O pins
– 2 analogue comparators
– I^{2}C interface
– Full duplex UART
– Serial in-circuit programmable

Because there are not enough I/O pins available for direct connection of the pushbuttons and the LCD in 8-bit mode, some pins have been given a double function by clever programming. Pushbutton connections '+' and '–' share a port pin line with the display control lines. For the display control these pins are programmed as push-pull stages, while input-only mode is briefly selected when polling the pushbuttons for activity. Resistors R1 and R2 limit the short-circuit current when a pushbutton is pressed at the same time the display is being controlled.

The light barrier receiver outputs are

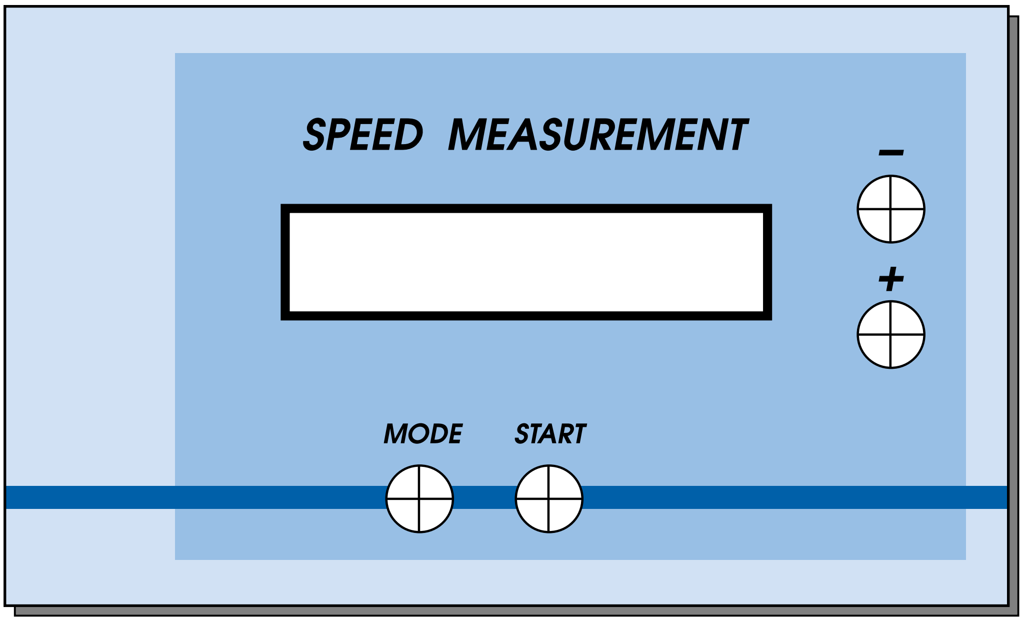


Figure 6. Suggested decal for the Heddic 222 case (not available ready-made).

directly connected to the microcontroller inputs. The internal pull-up resistors are used because the light barrier has an open-collector output.

Circuit and construction

The display is an alphanumeric dot matrix type with one line of 16 characters. This version should be widely available because it represents a kind of industry standard, including the use of the Hitachi HD44780 LCD controller and its command set. The display requires a single supply voltage of +5 V (no additional –5 V!). Important things to watch out for are the pinout and with it the order of the connections (upper left corner), as well as the internal RAM address allocation. Only LCD modules with a multiplex rate of 1/16 have the right address order 00, 01, 02, 03, 04, 05, 06, 07, 40, 41 to 47. Although more information has to be displayed than can be fitted on a single line, a 1-line display is used instead of a 4-line type because the user is prompted to press a button to view the information. The display contrast is adjusted by preset P1. Turn P1 across its full travel if no characters are displayed when the circuit is first started and you are sure that no construction errors have been made.

As usual the circuit is powered via a 7805 fixed voltage regulator.

With simplicity and cost in mind, the sender and receiver are connected to the processing electronics via a single 9-way sub-D connector. The light barrier wires have to be connected in accordance with Figure 3.

In this application, the sender control inputs are not used — they are either not connected or hard-wired to ground.

The circuit can be built in a jiffy using the circuit board whose design is shown in **Figure 5**. The display, pushbuttons and on/off switch are mounted at the underside of the board. The microcontroller is inserted into a socket.

The suggested enclosure for the project is a Heddic Type 222, which, being transparent, obviates the need to cut a clearance for the LCD face. The (suggested) front panel decal shown in **Figure 6** was designed with the Type 222 case in mind and should help to give the instrument a professional appearance.

Practical use

To enable the controller to respond instantly to changes at the light barrier outputs, the measurement routine is interrupt-driven. As a consequence, port pins P1.3 and P1.4 are not constantly polled by the software. Instead, a piece of logic inside the microcontroller is set up to control the time measurement. The advantage of this arrangement is that in addition to instant reaction to events, the software has sufficient spare time for other chores like driving the display and scanning the pushbuttons for activity. This is not possible just like that and using any input pin — P1.3 and P1.4 are 'specially prepared' to handle such exacting jobs.

At a falling pulse edge, the internal program halts instantly and jumps to a special routine written to handle the task on hand.

Any speed measurement starts when the first light beam is interrupted. Instantly, the 16-bit counter inside the microcontroller is started at a count rate of 1 µs. Overflows that occur are added in an 8-bit register, allowing a maximum period of 16.777215 seconds to be measured. The object velocity, v, is then computed by the microcontroller using the simple equation

$$v = d / t$$

using the distance, d, between the barriers and the timer state, t.

Using the default settings, the software assumes a barrier distance of 0.1 m, a readout in m/s and 'continuous' as the measurement mode.

To simplify the calculations, the resolution is limited to two decimals, which should be sufficient in most if not all cases. If you require better accuracy, get out your pocket calculator and use the indicated counter value and the light barrier distance. You should, however, always take the 0.5-ms delay of the light barriers into account. Note, however, that this delay is about equal on both receivers so that should not be a problem.

More important than tweaking the numbers behind the decimal point (or comma, in this case) is to make sure that the senders and receivers are correctly aligned and properly secured, while their distance remains easily adjustable for the speed measurement you want to perform.

With the barriers installed, connected up and the software switched on, the software takes over, guiding you through a menu in which various options may be selected.

MODE key

Press this key to step through the four menu items, as follows

TEST – DISTANCE – MODE – SPEED – TEST

TEST causes the state of the light barrier receivers to be displayed. The message **OK** means 'high level measured' and indicates that the light beam from the sender is being picked up. A Low indicates that the beam is either interrupted or not properly aimed at the receiver. Before any measurement is started, both outputs should flag 'OK'. After a reset or after the circuit is switched on, this menu always appears first.

It should be noted that a unconnected light barrier receiver will also cause OK to be displayed, simply because of the internal pull-up resistors at the controller inputs. Therefore, as a 'security check', make a habit of interrupting the light beam manually while watching the display indication.

Another source of errors is the lobe shape of the light beam, which may reach the other receiver when the barriers are placed at a relatively small distance.

When **DISTANCE** is displayed, the value to be processed by the microcontroller is entered in centimetres. The default value is 10 cm. As a matter of course, you have to make sure that the displayed value is the real distance between the light barriers.

MODE allows you to select the measurement mode:
SINGLE performs just one measurement. When the barriers are triggered again, no new measurement is performed.
CONTINUOUS enables results to be refreshed by any new measurement. The current measured value is always overwritten.

SPEED indicates the measured velocity of the object. Three units are available to choose from: m/s, km/h and seconds. The maximum time is 16.777215 seconds. Exceeding this value causes ERROR to be displayed.

START key

This pushbutton is only read in SINGLE mode. It launches a measurement when neither of the light barriers is interrupted. The display then indicates READY. If a barriers fails to detect a signal, the software jumps to TEST, where the user can see what's wrong. Once the problem is solved, another press of the START key causes the SPEED or DISTANCE menu to appear.

+ and – keys

In traditional fashion these two keys are used to select different parameters in the individual menus. In the DISTANCE menu, for example, the keys allow values between 1 cm and 255 cm to be set up.

In the MODE menu, both keys have the same function, changing between SINGLE and CONTINUOUS.

In the SPEED menu, finally, the + and – keys allow you to choose between a readout in m/s, km/h or seconds (the latter for the convenience of pocket calculator users).

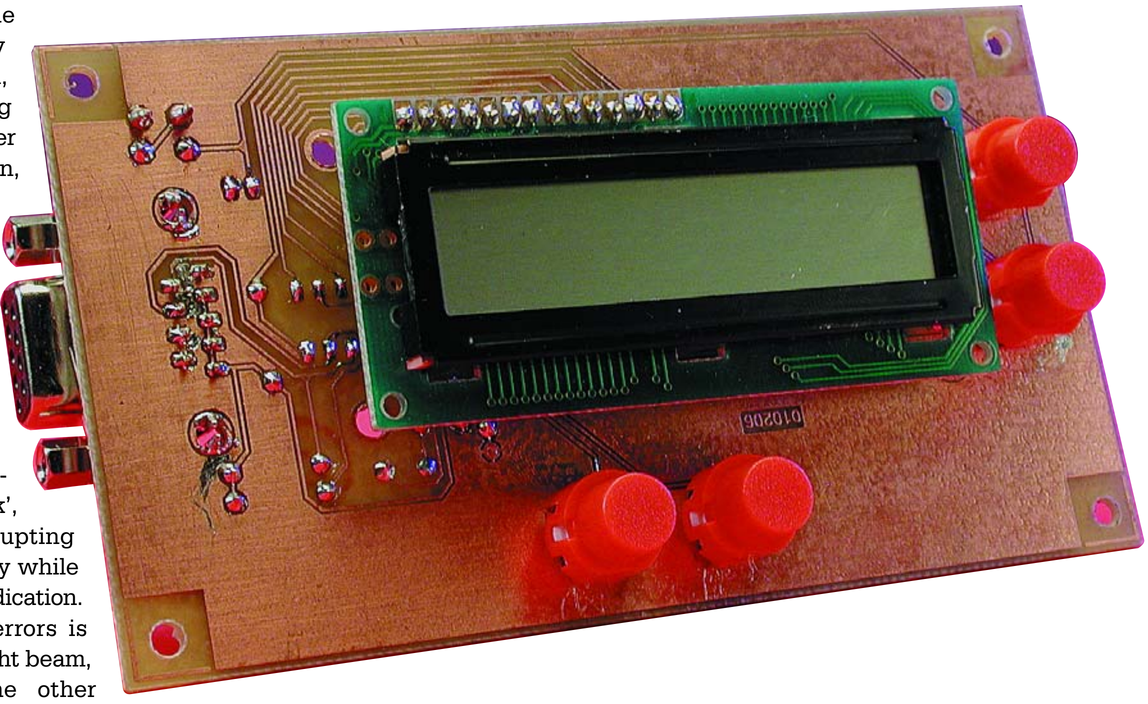
(010206-1)

Some useful addresses

Dot matrix display:
www.datamodul.de/displaytechnik/lcd-alpha/bt_11608.htm
http://electronicassembly.de/
www.schukat.com (Display Fa. Sharp)

Microcontroller:
www-us.semiconductors.philips.com/ mcu/

Light barrier datasheets:
www.kodenshi.com/pdfs/g-1.pdf



Remote Process Control using a Mobile Phone (2)

part 2*: programming and use

The SMS Chip checks incoming text for a password, switches outputs interactively or at preset times, controls an LCD and sends status information back to any mobile phone.

In the first part of this article we looked at the circuit design and hardware for this project and now we turn our attention to the PCB layout, the software and the text message structure.

A look at the component placement diagram in **Figure 1** shows that you will not need any special soldering skills to complete the circuit board. No wire links are required and all the ICs (except the voltage regulator) are fitted into sockets. All external connections are made via pinheaders or connectors mounted around the edge of the PCB. The LED arrays D1 and D2 can be replaced by groups of eight individual LEDs but if you use high-efficiency types it will be necessary to substitute 1 kΩ or 1.5 kΩ resistor networks for R1 and R2. In all cases it will still be necessary to use the ACT type buffer specified in the parts list for IC3 and IC4.

The voltage regulator IC8 requires a heat sink fitted along the top surface of the PCB. Be sure to fit an insulation sheet between the PCB and the heat-sink to ensure that none of the PCB tracks are short-circuited.

The on/off switch S4 can be simply mounted on the PCB or to a front panel if the unit is mounted in an enclosure.

Three screws with pillars are used to fix the LCD onto the SMS ExBo interface board. The fourth mounting hole does not line up with the hole in the interface board but a screw and pillar should be fitted to the LCD to give it additional support. Leave the bottom end of the pillar free. Non-conducting washers are used under the heads of the

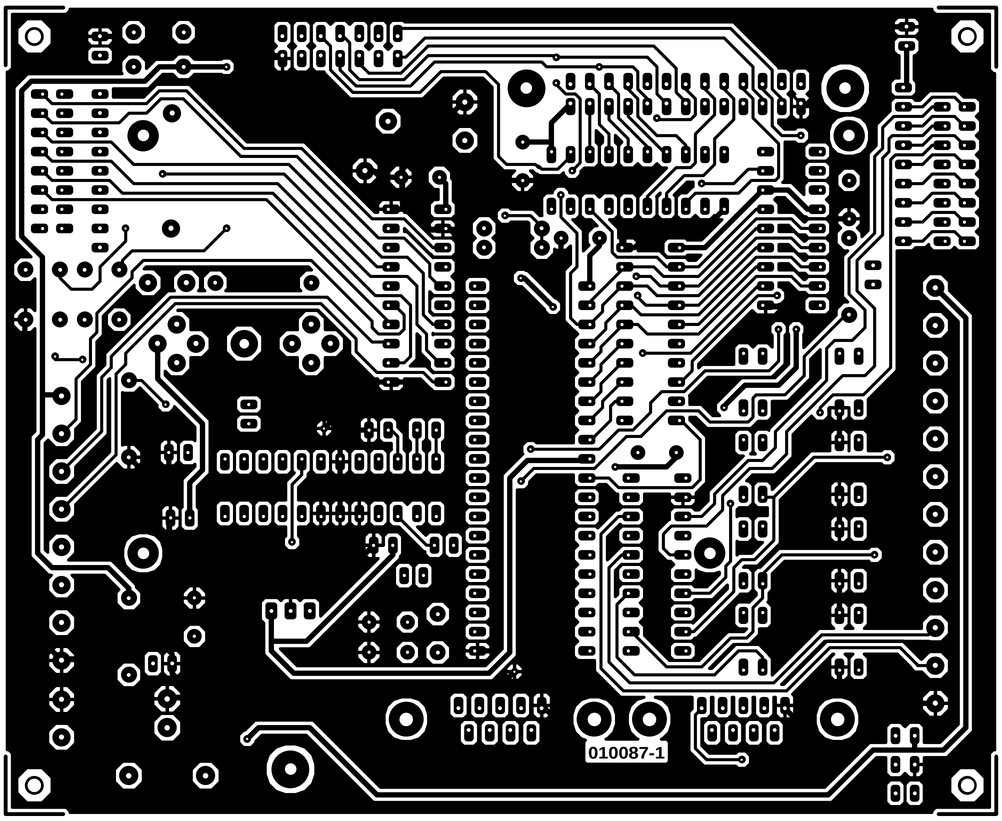
SMS Chip features

The SMS Chip* is implemented in an AT89LS8252 from Atmel (a programmable 8051 derivative)

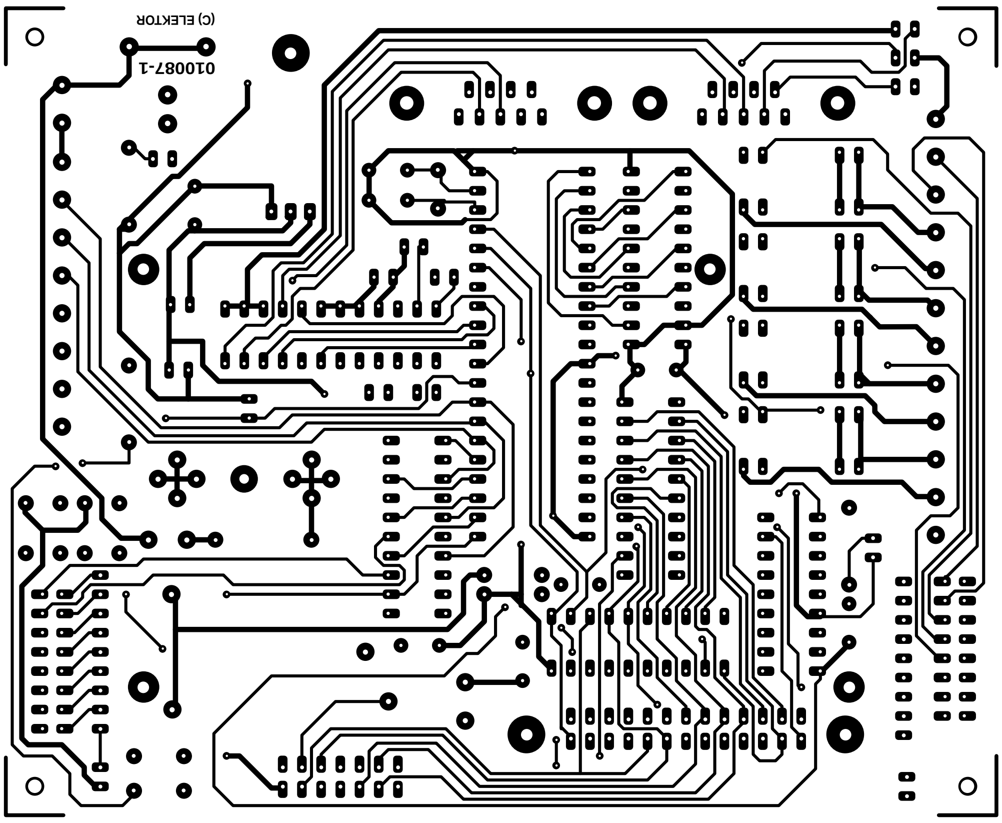
- 8 kByte On-Chip Flash-Programmable memory
- 2 kByte On-Chip Flash-Data memory
- 16 digital port pins (Port P1 and Port P2), individually programmable as inputs or outputs.
- Serial Interface 1 for communication with the mobile and loading the configuration data.
- Serial interface 2 UART implemented in software (9600 baud) for communication with an external μC system (PLC, PC)
- Software Real Time Clock (RTC) without battery back-up.
- Optional external hardware Real Time Clock (RTC) with battery back-up (IC6).
- Control for an alphanumeric LC Display to display text messages.
- Connector for external circuit interface (Port P0).
- Package outline DIP-40 or PLCC-44

* The SMS Chip is exclusively available from:
J. Engelmann & U. Schrader
Im Schmiedehofe 14
D-31035 Despetal-Barfelde
Germany.
Tel. (+49) 5182 903520
Fax (+49) 5182 903530
Website: www.engelmann-schrader.de
Email: info@engelmann-schrader.de

* This instalment could not be included in the February 2002 issue as planned because of late delivery of the German original, and the extra time needed to localize various project related files for use in English speaking countries. Our apologies for any inconvenience caused. *Ed.*



Component side



Solder side

Figure 1a. PCB layout and …

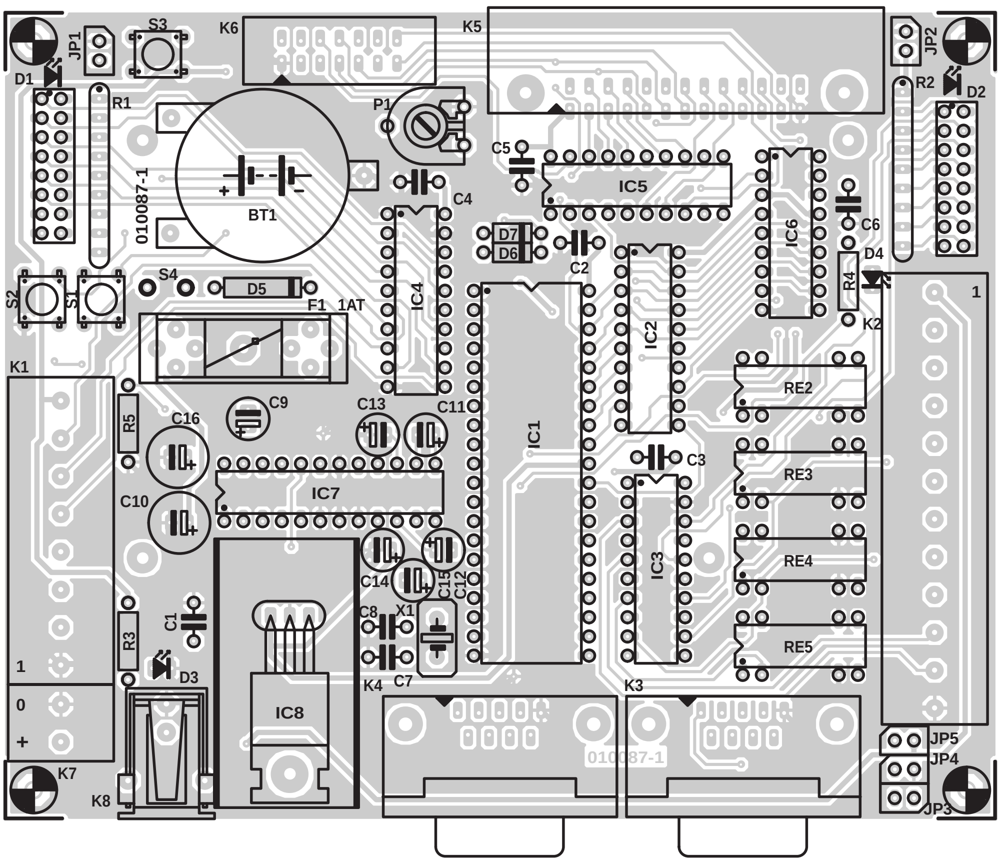


Figure 1b. … component placement of the SMS ExBo.

three screws fixing the pillars to the interface board to prevent possible short-circuits between any of the PCB tracks.

The Data Cable

Connection of the adapter card to a mobile can be accomplished using an off-the-shelf data cable for the Siemens '35 series GSM phone. These cables are usually used to connect the mobile to a PC serial port. Using such a cable has one big disadvantage in our application because it does not allow the mobile to be re-charged from the SMS ExBo. No provision is made in the cable to supply a charging current for the mobile's battery. If you used this cable then it would be necessary to periodically disconnect the mobile from the SMS ExBo and charge up the batteries with the phone charger unit. Using this option it is essential that both jumpers J4 and J5 are removed and jumper J3 fitted (see **Table 2** in the first part of this article). The preferred alternative is to make a slight modification to the data cable wiring. Firstly dismantle the plug housing at the phone end of the data cable (see Figure 2) and solder an additional wire (same length as the data cable) to pin 3. Lead this wire out through the cable entry opening and then re-assemble the connecter. Fit an in-line 1 Ω, 1 W resistor in this wire and connect the free end of the wire to + 5 V on the SMS ExBo (pin 1 of connector K2 would be suitable).

The charging current can be up to 500 mA so it is important not to stint on the mains adapter unit specified in the parts list. Fit jumper J3 but do not fit jumpers J4 and J5, these are only intended for future use with other types of mobiles.

SMS Chip operation

With the PCB finished and the data cable made up we can turn our attention to the SMS chip and how the SMS ExBo is controlled. The SMS Chip must first be initialised before it will recognise any commands sent in the SMS text.

SMS Chip initialisation

The SMS chip needs to be configured with information such as the mobile telephone number, pre-loaded text messages and so on before it can be used. The configuration data can be entered using a standard text editor program. See the accompanying text box giving an example of the configuration data.

The configuration data is stored as ASCII in the file `SMSCHIP.CFG` and the program used to send this data to the SMS ExBo is the DOS program `SMSCONFG.EXE`. To load configuration data it is first necessary to remove jumper J3 (jumpers J4 and J5 should not be fitted) and then fit a standard serial (female to female) 'null modem' cable from COM1 of the PC to K3 of the SMS ExBo. Press the reset button S3 and the 'mobile ready' indicator (D4) labelled 'power' on the circuit diagram will flash to show that the SMS Chip is waiting 10 s for the configuration data to be sent. Two clicks on the `SMSCONFG.EXE` file will automatically open up a DOS window and show transfer of the configuration data.

Some users may need to use the COM2 port, in which case you should call the program as before from the DOS window, but this time use the command

COMPONENTS LIST

Resistors:
R1,R2 = 8-way 330Ω SIL array
R3 = 1kΩ5
R4 = 1kΩ8
R5 = 4kΩ7
P1 = 10kΩ preset

Capacitors:
C1, C2, C4,C5,C6 = 100nF (5mm lead pitch)
C3 = not fitted
C7,C8 = 27pF
C9 = 10µF 63V radial
C10 = 100µF 25V radial
C11-C15 = 1µF 16V radial
C16 = 100µF 10V radial

Semiconductors:
D1,D2 = LED, 3mm (8 pcs) or array *
D3, D4 = LED, green, high efficiency
D5 = 1N4002
D6 = BAT48
D7 = 1N4148
IC1 = AT89(L)S8252-24PC in DIP40 case. Programmed, available from Engelmann&Schrader
IC2 = 74HCT573
IC3,IC4 = 74ACT240*
IC5 = GAL16V8, programmed, Publishers' order code **010087-31**
IC6 = RTC72421
IC7 = MAX207 or ADM207EAN
IC8 = 7805 with ICK35SA heatsink *

Miscellaneous:
BT1 = CR2032 Lithium button cell with PCB mount holder
F1 = fuse, 1A, slow, with PCB mlunt holder
JP1-JP5 = jumper
K1 = 8-way PCB terminal block, lead pitch 5mm
K2 = 12-way PCB terminal block
K3,K4 = 9-way sub-D plug (male), PCB mount, angled pins *
K5 = 26-way boxheader, angled pins
K6 = 14-way boxheader
K7 = 2-way PCB terminal block, lead pitch 5mm
K8 = mains adaptor socket, PCB mount
RE1 = not fitted
RE2-RE5 = 5V PCB mount relay, 1 make or 1 changeover contact, with flyback diode (e.g., Meder 1A72-12D 5V or Siemens V23100-V4305-C010 or Conrad Electronics #504580) *
S1,S2,S3 = pushbutton with 1 make contact
S4 =switch, on/off *
X1 = 11.0592MHz quartz crystal
LC Display = alphanumeric display with HD44780 controller (or compatible), e.g., 4 lines of 20 characters
PCB, Publishers' order code **010087-1**
Disk, project software, Publishers' order code **010087-11**
Battery eliminator (mains adaptor) 9V / 1A

* See text

```
SMSCONFG.EXE /2
```

to start the program. Once the transfer has been completed, close the DOS window and press reset button S3. D4 will again blink for 10 s (it has already been configured so just ignore D4) and then the SMS Chip looks for a mobile connected to K3. Disconnect the null-modem cable from the PC to the SMS ExBo and connect the mobile to K3 using the data cable. Jumper J3 can now be fitted.

A short description of the configuration data file together with valid parameters is given in a text box in this article but for a more detailed description refer to the SMS Chip handbook.

The SMS Chip instruction set

After the SMS Chip is configured it is ready to accept commands. Altogether there are twelve commands that the SMS Chip recognises and these can easily be entered using SMS 'texting' (see **Table 1**). All of the commands have the same structure: first comes the password then the command and finally the message terminator. Detailed descriptions of all the commands are included in the handbook for the SMS Chip, free to download from the *Elektor Electronics* web site. Any visitor to the web site will also notice an additional program associated with this project written in C51. The SMS-1.c and its associated hex file SMS-1.hex. This program allows anyone who has built the 537 'Lite' computer featured in the January/February 2000 editions of Elektor Electronics to use it as an external computer at the remote site. In this configuration the

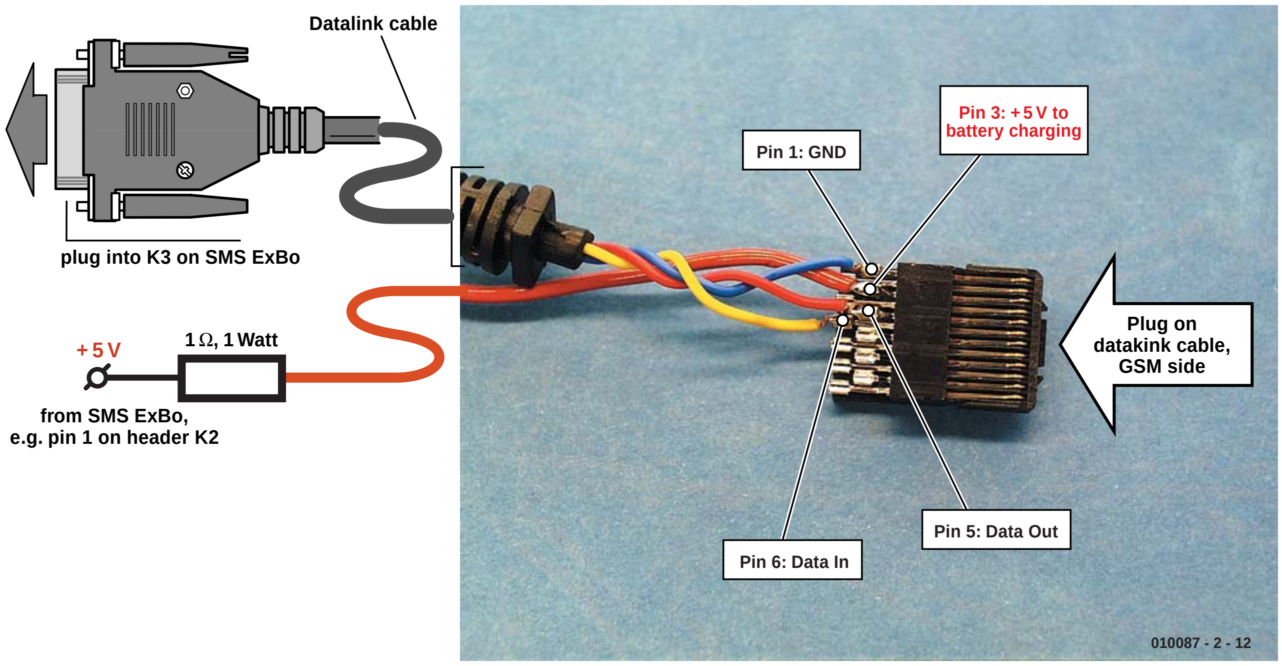


Figure 2. Modification of an off-the-shelf data cable to allow in-circuit charging of the mobiles battery.

Free Software

To download any of these files go to the Free Downloads area of our website at www.elektor-electronics.co.uk

SMSCHIP.CFG	Configuration data
SMSCONFG.EXE	load program for smschip.cfg
SMS-1.c	C control program for the 537 'Lite' computer
SMS-1.hex	Hex file for the above program sms-1.c

537 'Lite' computer will receive SMS text from the SMS ExBo and send its own text message back out through the same path.

Putting it all to use

When a message is received the SMS Chip reads all the SMS text and then automatically erases the message from the mobile's memory. If a standard text message (without any password or configuration commands) is sent to the mobile, the SMS Chip will try to decode it but find nothing of interest in the message so it will just delete it from the mobiles message store.

An important feature of this system is the response time of messages through the GSM network. The time delay between sending a message from one mobile and it being received on the other mobile can vary from a few seconds up to an hour or so. It is therefore important to note that this system will not be suitable to control or monitor time-critical processes! Suitable applications would be for example controlling heating systems or monitoring cold-storage refrigerators.

During set-up and testing you will need to send many messages to the system just to confirm correct operation of the unit. A good alternative to using a mobile phone to send the messages is to use an Internet connected PC. There are several web sites that allow you to send text messages to mobile phones free of charge (try www.lycos.co.uk). This will save not only the strain on your finances but also help avoid that painful medical condition known as 'texters finger'. These web sites add an advertising slogan either before or after your message but the SMS chip will simply ignore it. It only decodes text following the 'message start' marker (the password) and stops decoding when the 'message end' symbol (close bracket) is received. All of the text outside these markers will be ignored. A problem could occur if the advertising slogan has been inserted in the middle of your message but this is very unlikely and I've never yet found a site that does this. If this were to occur, the SMS chip would recognise that the message has been corrupted and ignore the message completely. A valid message is necessary before any of the outputs are switched.

If you object to the advertising slogans then it will be necessary to find a site that does not use them. However in the world of the non-existent free lunch (TANSTAFL principle) you will have to pay for sending the text. Other sites on the Internet offer an e-mail to SMS gateway. As the name suggests these sites provide a link between the GSM text network and the Internet so that e-mails can be sent and received as text messages on a GSM phone (www.locust.co.uk or www.airmail.co.uk). Additional information on this option can be found in the handbook.

(010087-II)

Table 1. The SMS Chip command instruction set.

```
(
"<password>"

SET             <pin>                                        // Set the specified pin to 1
 or
SET             <identifier>                                 // Set the specified pin to 1

RESET           <pin>                                        // Reset the specified pin to 0
 or
RESET           <identifier>                                 // Reset the specified pin to 0

OFF             <pin>                                        // Set the specified pin to 1
 or
OFF             <identifier>                                 // Set the specified pin to 1

ON              <pin>                                        // Reset the specified pin to 0
 or
ON              <identifier>                                 // Reset the specified pin to 0

PULSE           <pin> <duration>                             // Output a pulse on the specified pin
or
PULSE           <identifier> <duration>                      // Output a pulse on the specified pin

OUTPUT          <address> <values>                           // 8-Bit output from XDATA (8 Bit Address)

INPUT           <address> <count>                            // 8 bit input from XDATA (8 Bit Address)

WRITE           "<text>"                                     // Output to the 2. serial interface

DISPLAY CLS                                                  // Clear LC Display
 or
DISPLAY SCROLL  "<text>"                                     // Enter Text in the last display line and scroll
 or
DISPLAY         <column> <line> "<text>"                     // Enter text giving line and column information

EVENT           <time> DELETE                                // Delete time EVENT
 or
EVENT           <time> <address> <count> SINGLE              // Time EVENT occurs once only
 or
EVENT           <time> <address> <count> EVERYDAY            // Time EVENT occurs daily

TIME            <time>                                       // Current time

REPORT NO                                                    // No report (Default)
 or
REPORT YES                                                   // report
 or
REPORT ERROR                                                 // Report only when an error occurs
)
```

Configuration file example.

```
-CONFIG
Time            :1200
Name            :Enterprise
Password        :Scotty
Masterno        :441794711081
Pin  0 I/O      :I
Pin  0 Name     :Warp1
Pin  0 Telefon :441781234567
Pin  0 Text     :Pressure too high!
Pin  1 I/O      :I
Pin  1 Name     :ALARM
Pin  1 Telefon :4512345876
Pin  1 Text     :Romulan on board!
...
Pin  7 I/O      :I
Pin  7 Name     :Pin7
Pin  7 Telefon :441704711081
Pin  7 Text     :Report pin 7!
...
Pin  8 I/O      :O
Pin  8 Name     :Pin8
Pin  8 Telefon :440000000000
Pin  8 Text     :(Pin8)
...
Pin 15 I/O      :O
Pin 15 Name     :Photon24
Pin 15 Telefon :440000000000
Pin 15 Text     :(Pin15)
Event  1 Time   :1200
Event  1 Typ    :E
Event  1 Start :00
Event  1 Count :05
Event  2 Time   :1205
Event  2 Typ    :E
Event  2 Start :10
Event  2 Count :12
Event  3 Time   :1210
...
Event  8 Time   :0000
Event  8 Typ    :O
Event  8 Start :00
Event  8 Count :0
LCD Lines       :04
LCD Char/Lines :20
LCD Addr 0      :00
LCD Addr 1      :64
LCD Addr 2      :20
LCD Addr 3      :84
```

Data header

-CONFIG
Key word indicating start of the configuration data.

Time
Actual time for the software Real Time Clock (RTC) in the SMS chip. Note: no separating colon is entered between hours and minutes.

Name
This allows an identifying name to be allocated to the SMS chip. It will use this name whenever it sends out SMS text.

Password
Every SMS text message sent to the remote mobile must use the password defined here. Any SMS message received without this password will be ignored by the SMS chip.

Masterno
This is the main telephone number that the SMS chip will send its text messages to.

Digital I/O port pins

Pin x I/O
This defines pin x as an input or output: I ≡ Input, O ≡ Output. When the pin is configured as an input, the SMS chip will automatically send out a message when the input level changes from high to low (falling edge). It will only send another message when the input level returns to a high state

Pin x Name
This allows you to assign a name to a pin so that in the text message you can refer to the pin number or the pin name whichever is more meaningful.

Pin x Telefon
The mobile telephone number entered here will be called and will receive the SMS text when a falling edge is detected on pin x. If no number is entered then the mobile with the master number (Masterno) will be called.

Pin x Text
The message (32 characters max) that will be sent out when the change in pin x is detected.

"Pin x Telefon" and "Pin x Text" are only valid if pin x is defined as an input pin.

Internal and timed events

The SMS chip can send system information automatically at pre-defined times to the mobile phone number (Masterno). A total of eight events are available (x=1 to 8).

Event x Time
The time that the SMS message will be sent.

Event x Typ
Three different types of event are possible:
O (Off): The corresponding event is deactivated.
S (Single): The event will be activated once only at the time specified.
E (Everyday): The event will be activated daily at the time specified.

Event x Start
The start address (decimal) in external memory of data (XDATA range) that will be included and sent out in the SMS text message.

Event x Count
This defines how much data and the start address of the data to be sent out. In the example listing event 2 will send out 12 data bytes in an SMS message from the external memory starting at address 10 at 12:05 every day.

The use of external memory is not covered in this article but the SMS ExBo system bus is available at connector K5 for the user to attach external memory (e.g. dual port RAM) to store this data.

LCD operation

LCD Lines
The number of lines on the LCD.

LCD Char/Lines
Characters per line

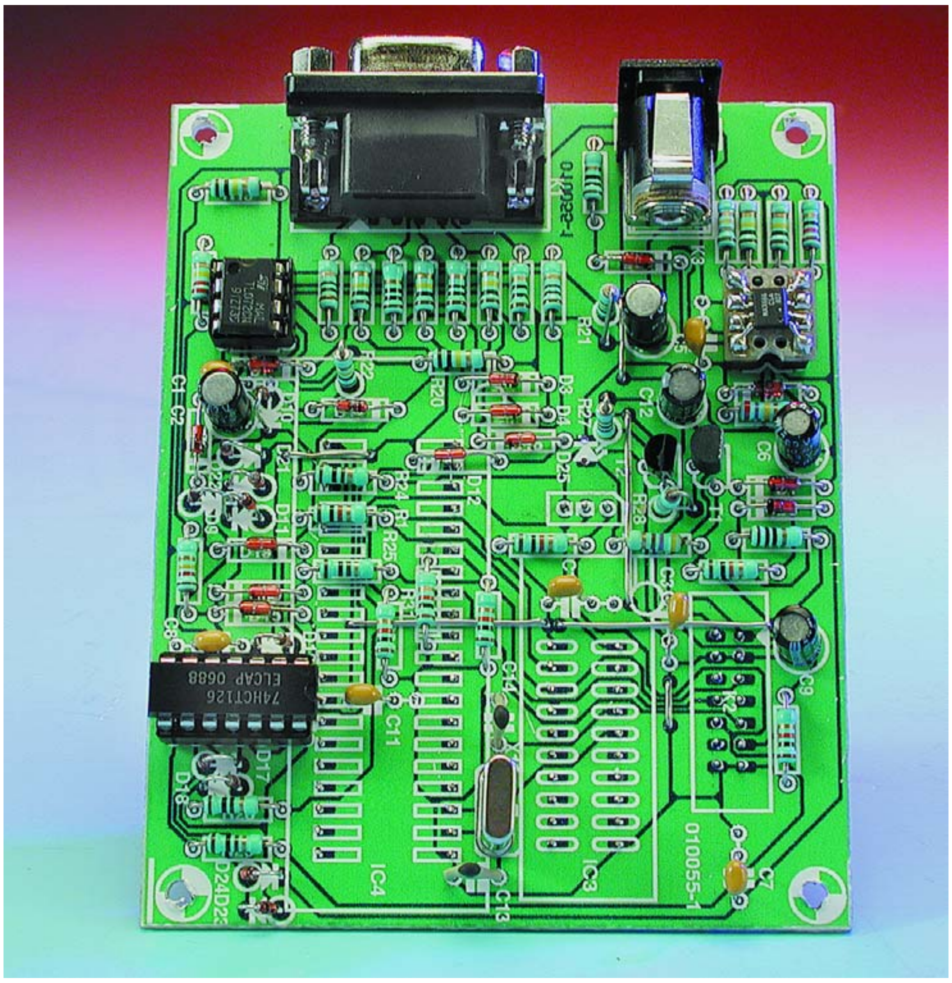
LCD Addr 0 to 3
The address of the first character in each of the display lines. The values are given in the corresponding data sheets.

Simple AVR Programmer

for (almost) all AVR microcontrollers

Design by H.-J. Hanft

Thanks to their high performance, Flash program memory, integrated hardware functions and low power consumption, Atmel AVR microcontrollers are becoming increasingly popular, even in the semi-professional area.



The AVR series of microprocessors is based on Flash program memory and thus can be easily programmed using an SPI interface. The ability to reprogram the program memory (up to 1000 write cycles) makes these devices very attractive for use in the semi-professional area.

The documents and tools that you need for programming these microcontrollers, such as an assembler and debugger, can be obtained free of charge from the Internet site www.atmel.com. These tools are for use with the Windows operating system (95/98/NT). You won't find any development tools that run under MS-DOS here, which is a good reason for requiring the programmer and associated software to work under Windows.

This simple programmer provides an extremely economical starting point for developing applications that use AVR microcontrollers. In designing the AVR programmer, special attention was given to achieving an economical, simple and robust construction that can cope with adverse ambient conditions, such as electrostatic discharge, short circuits and induced noise.

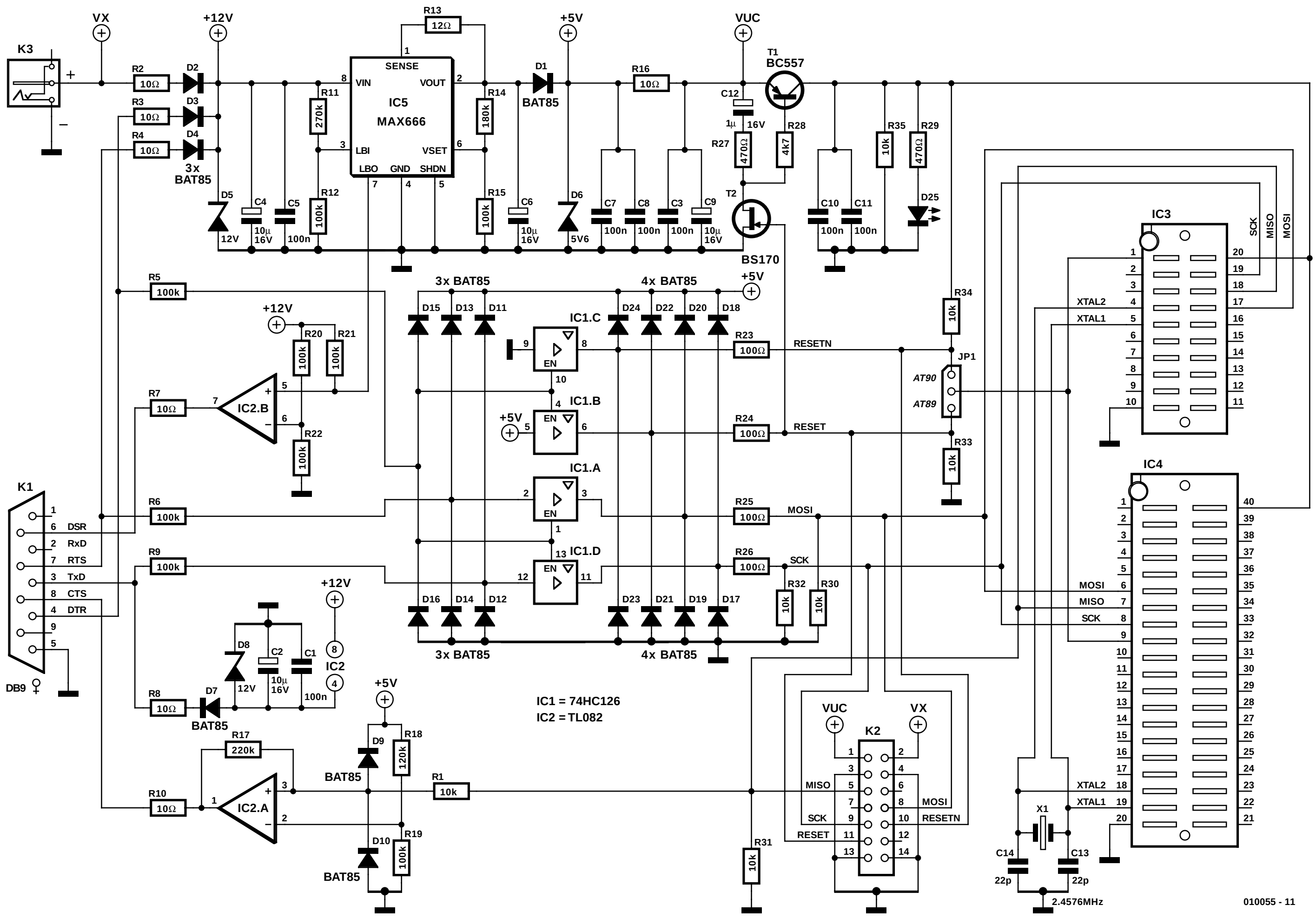


Figure 1. Instead of an interface converter, two opamps and a logic gate provide the level conversions between the PC and the microcontroller.

The programmer is connected to the computer via an RS232 interface using a 1:1 RS232 cable (not a null modem!). If you only want to program the Flash memory, no external supply voltage is necessary for using the equipment. The supply current (a few milliampères) needed to program a single microcontroller can be drawn directly from the serial interface of the computer.

The programmer can also be used as an in-circuit programmer for programming a microcontroller that is already built into a circuit. In this case, under certain conditions it will be necessary to use an external power supply.

The universal programming software is implemented as a console application for Windows 95, 98, NT, ME, 2000 and XP.

Features

The special features of the programmer are:

– simple, economical and robust construction
– universal SPI interface, also suitable for other applications
– programming speed up to 256 baud (depends on PC)
– can also be used for in-circuit programming
– no external power supply required
– 'low voltage' indicator warns of insufficient supply voltage
– all SPI outputs are short-circuit proof and have overvoltage protection
– maximum regulated output current with an external power supply: 30 mA
– all SPI inputs and outputs have automatic level adaptation
– insensitive to electrostatic discharges and induced noise

Voltages à la carte

The essential tasks of the hardware of the AVR programmer are generating the voltages necessary for programming and converting signals from the microcontroller to RS232 levels and vice versa. The positive supply voltage for the AVR programmer is taken from the two signal lines RTS and DTR on the computer's RS232 interface (or from a mains adapter connected to K3). The voltage on the signal lines can range between –5 to –12 V or +5 to+12 V. These voltages are 'ORed' and rectified using diodes D1–D4 and limited to around 12 V by Zener diode D5 (just to be safe). Resistors R2–R4 provide current limiting and attenuate interference coming from the RS232 interface.

In practice, the maximum voltage appearing at the RS232 interface is normally significantly less than the limiting value of 12 V. However, if an external power supply is used, you must ensure that the supplied voltage does not exceed 12 V.

The 12-V supply voltage is only used to power the opamp. The remainder of the circuitry (IC1, a 74HC126) needs a stabilised 3.6-V supply voltage, which is provided by voltage regulator IC5 (MAX666). The output voltage of this regulator set to 3.6 V using the formula

$R14 = R15\ (V_{OUT} \div 1.30\ V - 1).$

Resistor R13 sets the maximum output current of the regulator according to the formula

$R13 = (0.5\ V) \div I_{CL}.$

Here the maximum current is around 30 mA. Since the maximum current that can be drawn from the RTS and DTR lines is around 10 mA, you can increase the value of the resistor to 33 Ω if you have no intention of using an external power supply.

The negative supply voltage for the opamp is taken from the TxD line, whose voltage in the quiescent state lies between –5 V and –12 V and is thus particularly suitable for this purpose. The negative voltage is

COMPONENTS LIST

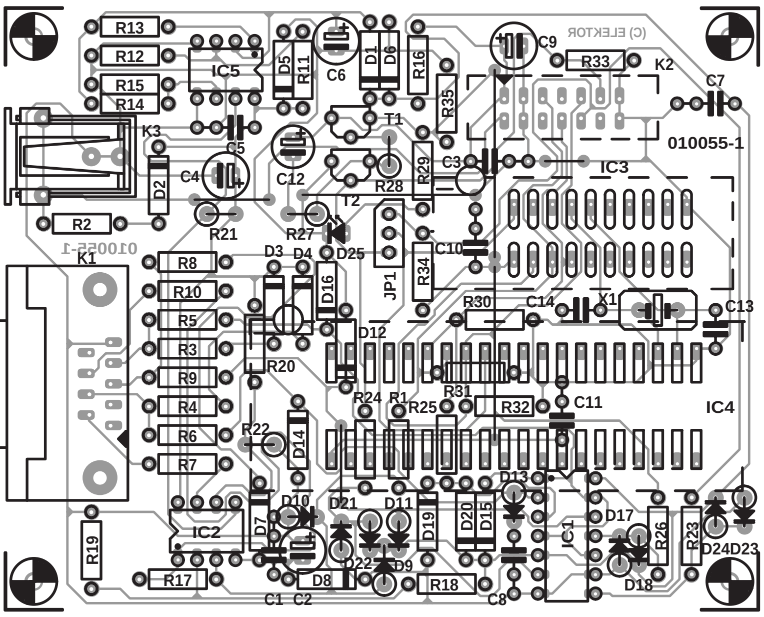
Resistors:
R1,R30-R35 = 10kΩ
R2,R3,R4,R7,R8,R10,R16 = 10Ω
R5,R6,R9,R12,R15,R19-R22 = 100kΩ
R11 = 270kΩ
R13 = 12Ω
R14 = 180kΩ
R17 = 220kΩ
R18 = 120kΩ
R23...R26 = 100Ω
R27,R29 = 470Ω
R28 = 4kΩ7

Capacitors:
C1,C3,C5,C7,C8,C10,C11 = 100nF
C2,C4,C6,C9 = 10µF 16V radial
C12 = 1µF 16V radial
C13,C14 = 22pF

Semiconductors:
D1-D4,D7,D9-D24 = BAT85
D5,D8 = zener diode 12V 500mW
D6 = zener diode 5V6 500mW
D25 = LED, red, high efficiency
T1 = BC557
T2 = BS170
IC1 = 74HC126
IC2 = TL082
IC3 = 20-way ZIF socket
IC4 = 40-way ZIF socket
IC5 = MAX666CPA or -EPA

Miscellaneous:
JP1 = 3-way pinheader with jumper
K1 = 9-way sub-D plug (male), angled pins, PCB mount
K2 = 14-way boxheader
K3 = mains adaptor socket
X1 = 2.4576MHz quartz crystal
PCB, order code **010055-1**
Disk, project software, order code **010055-11**
(see Readers Services page)

PCB layout and project software available as free downloads from www.elektor-electronics.co.uk



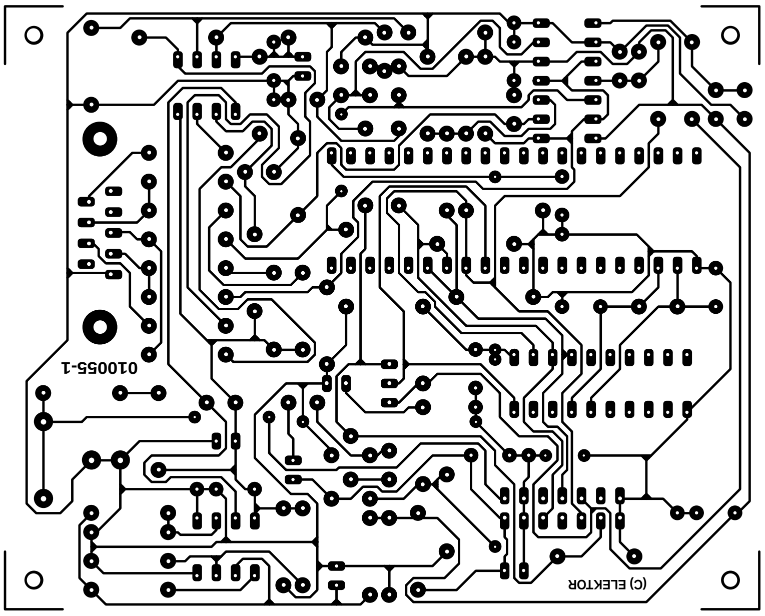


Figure 2. All components are located on a single-sided printed circuit board (available ready-made) The two ZIF sockets, the jumper, the LED and the in-circuit-programming socket belong on the bottom side.

limited to –12 V by D7.

The MAX666 measures the input voltage via the voltage divider R11/R12. The output of the voltage monitor, pin 7 (LBO = Low Battery Output), becomes active if the voltage given by the formula

$R11 = R12 (V_{BATT} \div 1.30 \text{ V} - 1)$

drops below the minimum value. This signal (open drain) is converted to an RS232 signal (DSR) by opamp IC2b and evaluated by the software for the AVR programmer. The software ensures that an adequate voltage is always available. An external power supply may be needed for programming EEPROMs (depending on the PC used), but it is not necessary for programming AVR microcontrollers.

When you are handling the microcontroller, the stabilised voltage from the MAX666 must be disconnected from the socket. This task is looked after by the software via the DTR line, with the help of the two switching transistors T2 (BS107) and T1 (BC557). RC network R27/C12 delays switching off the voltage in order to maintain the voltage needed to erase the chip (Chip Erase) during the Reset pulse. Resistor R35 quickly discharges the residual voltage once the supply voltage for the microcontroller has been switched off.

From TTL to RS232 and vice versa

The microcontroller TTL signal MISO is converted to an RS232 level and is then called CTS. Here, instead of the usual MAX232, we use an opamp (IC2a) wired as a comparator, exactly as for the previously described conversion of LBO to DSR. This is not only less expensive, it also allows specific reference (threshold) voltages and a switching hysteresis to be set. For in-circuit programming in particular, this quite important, since the AVR microcontroller supply voltage provided by the target circuit can vary over a range of 2.7 to 6.0 V. The relevant formulas are

$U_{on} = U_{ref} - R1 / R17 (U_{iH} - U_{amin})$

$U_{off} = U_{ref} - R1 / R17 (U_{iL} - U_{amax})$

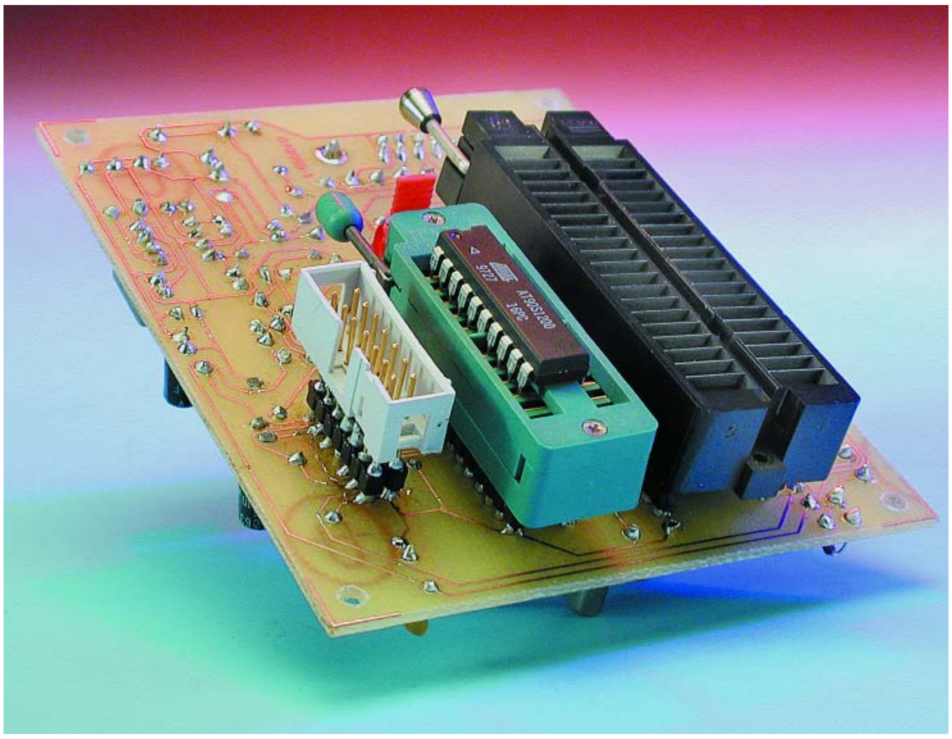


Figure 3. You can also make do with a 24-pin socket…

for the switch-on and switch-off levels and

$U_{ref} = V_{cc} R19 / (R19+R18)$

for the reference voltage. Naturally, the same thing applies to IC2b.

The conversion of the signals DTR, RTS and TxD into the corresponding signals RESET, MOSI and SCK is implemented using IC1 (74HC126). The four output signals (RESET, RESETN, MOSI and SCK) are enabled by a High level on the DTR line. Excessive voltages on the IC inputs are reduced to the proper level by two clamping diodes and a current-limiting resistor on each input.

For the driver outputs, two clamping diodes also provide overvoltage protection, which is of considerable importance when using a long flatcable for in-circuit programming.

As is only proper with a good programmer (and as shown by the component layout in **Figure 2**), all components are mounted on the printed circuit board, including all of the sockets for:

– a mains adapter,
– a 1:1 RS232 cable to the PC and
– in-circuit programming (K2).

Naturally, zero insertion force sockets for 20-pin and 40-pin DIL packages are provided for the microcontrollers to be programmed. They must be fitted to the bottom side of the circuit board. This also applies to K2, the LED and jumper JP1. 20-pin ZIF sockets are practically unavailable, but a 24-pin model works just as well, as can be seen in **Figure 3**. The extra pins can simply be allowed to 'dangle' in the air.

The circuit board is printed on only one side (which makes it less expensive), so several wire bridges are necessary. Fitting the remainder of the components to the board does not need any explanation.

AVR programmer software

The software includes all of the settings and functions necessary for serial programming of the microcontroller:

– choosing the COM interface, programming mode and programming speed
– reading, writing and verifying the program memory (Flash memory)
– reading, writing and verifying the integrated EEPROM (not present in all AVR microcontrollers)
– erasing the memory (Flash and EEPROM)
– setting the Lock bit

The settings and functions can be specified either directly as command parameters or as parameters in a command file (AVR.PROG.COM). The file generated by the AVR assembler or compiler in the Atmel-specific ‘Generic Hex Code’ can be used directly to program an AVR microcontroller. Table 1 lists the parameters needed for programming.

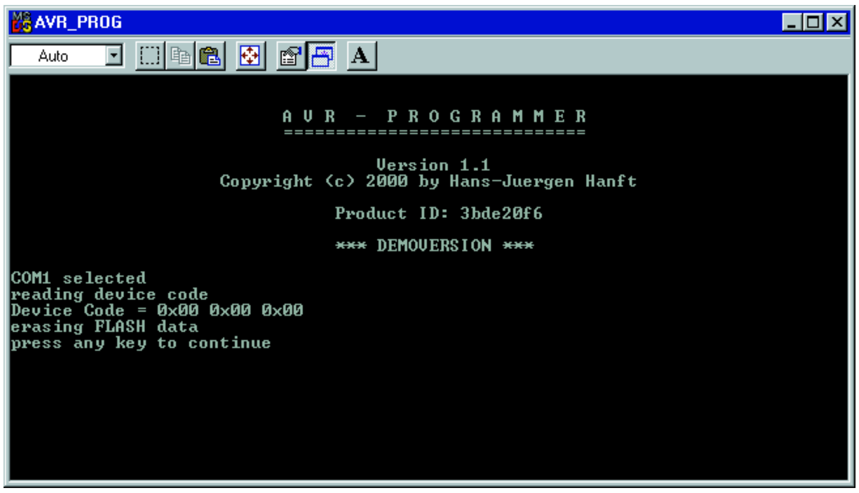
The following is an example of calling the AVR programmer software in a command line:

```
C:\AVR_PROGRAMMER/AVR_PROG
/COM1 /WRF(dice.hex,,)
```

This causes the file `dice.hex` to be programmed into the Flash memory. Optionally, supplementary start and end addresses for the programming can be specified. However, this is not necessary unless only part the program memory is to be written.

Should the program memory already be written, the ERASECHIP function must be first executed. In

Function parameters	Description
/**RDF**(file name, start address, end address) /**RDFLASH**(file name, start address, end address) /**READFLASH**(file name, start address, end address)	Reads the Flash memory of the AVR microcontroller and stores the data in a file.
/**WRF**(file name, start address, end address) /**WRFLASH**(file name, start address, end address) /**WRITEFLASH**(file name, start address, end address)	Writes data from a file to the Flash memory of the AVR microcontroller.
/**VYF**(file name, start address, end address) /**VYFLASH**(file name, start address, end address) /**VERIFYFLASH**(file name, start address, end address)	Compares the data in the Flash memory of the AVR microcontroller with the data in a file.
/**RDEEP**(file name, start address, end address) /**RDEEPROM**(file name, start address, end address) /**READEEPROM**(file name, start address, end address)	Reads the EEPROM memory of the AVR microcontroller and stores the data in a file.
/**WREEP**(file name, start address, end address) /**WREEPROM**(file name, start address, end address) /**WRITEEEPROM**(file name, start address, end address)	Writes data from a file to the EEPROM memory of the AVR microcontroller.
/**VYEEP**(file name, start address, end address) /**VYEEPROM**(file name, start address, end address) /**VERIFYEEPROM**(file name, start address, end address)	Compares the data in the EEPROM memory of the AVR microcontroller with the data in a file.
/**RDDC** /**RDDEVICECODE** /**READDEVICECODE**	Reads the device code of the AVR microcontroller.
/**EC** /**ERASECHIP**	Erases the memories of the AVR microcontroller (Flash and EEPROM).
/**SLB1** /**SETLB1** /**SETLOCKBIT1**	Sets Lock Bit 1 of the AVR microcontroller.
/**SLB2** /**SETLB2** /**SETLOCKBIT2**	Sets Lock Bit 2 of the AVR microcontroller.
/**WAIT**(x)	Interrupts the processing of the commands for x milliseconds ($0 \leq x \leq 60000$).
/**BREAK**	Interrupts the processing of the commands. Pressing any key causes processing to resumes with the next command.
Setting parameters	**Description**
/**COM**x	Specifies the COM Port interface. x = 1–8 (default x = 1) /SCK(x)
/**SCK**(x)	Specifies the programming speed. x = 1–12 (default x = 7)
/**MODE**(x)	Specifies the programming mode. x = 0 → AT90S.... mode (default) x = 1 → AT89S.... mode
/**+MOSI**	Sets positive logic for the MOSI signal. (default)
/**–MOSI**	Sets negative logic for the MOSI signal.
/**+MISO**	Sets positive logic for the MISO signal. (default)
/**–MISO**	Sets negative logic for the MISO signal.
/**+RESET**	Sets positive logic for the RESET signal. (default)
/**–RESET**	Sets negative logic for the RESET signal.
/**+TESTVCC**	Enables verification of the supply voltage prior to programming. (default)
/**–TESTVCC**	Disables verification of the supply voltage prior to programming.



```
              A V R  -  P R O G R A M M E R
              ============================

                       Version 1.1
         Copyright (c) 2000 by Hans-Juergen Hanft

                  Product ID: 3bde20f6

                  *** DEMOVERSION ***

COM1 selected
reading device code
Device Code = 0x00 0x00 0x00
erasing FLASH data
press any key to continue
```

Figure 4. The software (Windows console application) for the programmer.

this case, the command line to call the program looks like this:

```
C:\AVR_PROGRAMMER/AVR_PROG /COM1
/EC /WRF(dice.hex,,)
```

It is also possible to specify supplementary start and end addresses for data comparison. This results in a function call having the following form:

```
C:\AVR_PROGRAMMER/AVR_PROG  /COM1
/VYF(dice.hex,0x20,0x2F)
```

The microcontroller EEPROM can be programmed in a similar manner.

(010055-1)

IEC 1107 Electricity Meter Interface

talk to your electricity meter

By C. Mester

Many electricity meters installed since the mid 1980's are equipped with an IEC1107 compliant optical interface. This gives a convenient method for the Electricity Company to access all manner of information held in the meter using a hand terminal. This simple project describes a neat IEC1107 to RS232 interface that together with some software allows a PC to talk to the meter and retrieve energy consumption information[1].

[1] *Editor's note:* Some IEC1107 compliant electricity meters used in the UK and other European countries may have password protection. The software referred to in this article does not make provision for passwords to be entered. (*Ed.*)

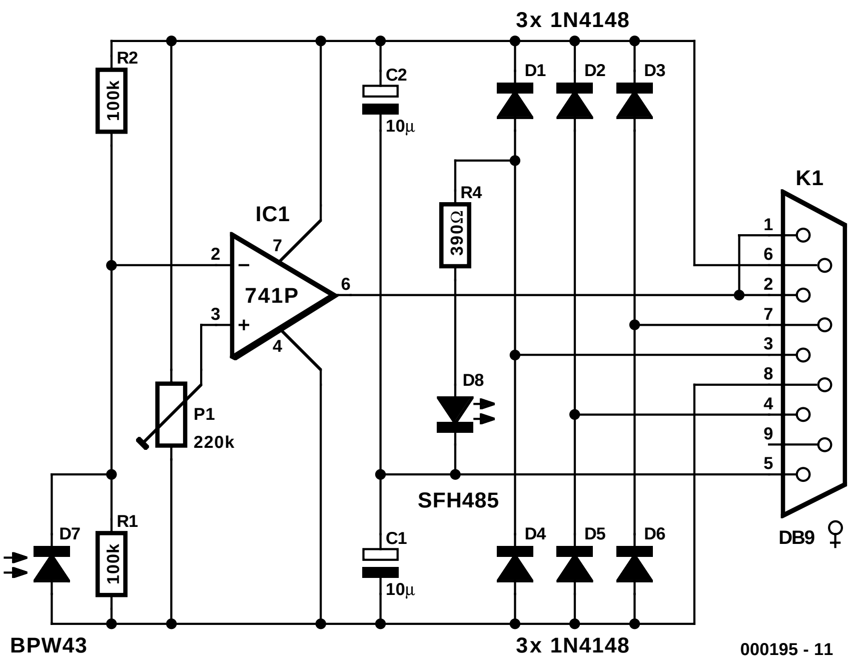


Figure 1. The ultra-simple interface circuit. Power for the transmitting diode is provided directly from the TXD signal. An Opamp comparator is used to amplify the received signal.

Among its prodigious output of documents describing international electrical standards the International Electrotechnical Commission (IEC) produced a specification IEC 1107 that defines a software communication protocol and the hardware necessary to pass information to and from electronic meters used to measure such things as electricity, gas and water consumption in the home and industry. Practically all the consumer meters manufactured these days have a microprocessor lurking somewhere inside and most are equipped with an IEC 1107 compliant interface or the equivalent European spec EN61107. The optical interface was originally introduced by the companies Ferranti and Landis & Gyr hence is alternative name, the FLAG port, derived from the company initials. This interface simplifies many functions of the meter operation and here in the UK it is possible for an operator from the electricity company to reset internal registers, alter the meter configuration and change tariffs via this interface. Before you get too excited I think we should point out that all

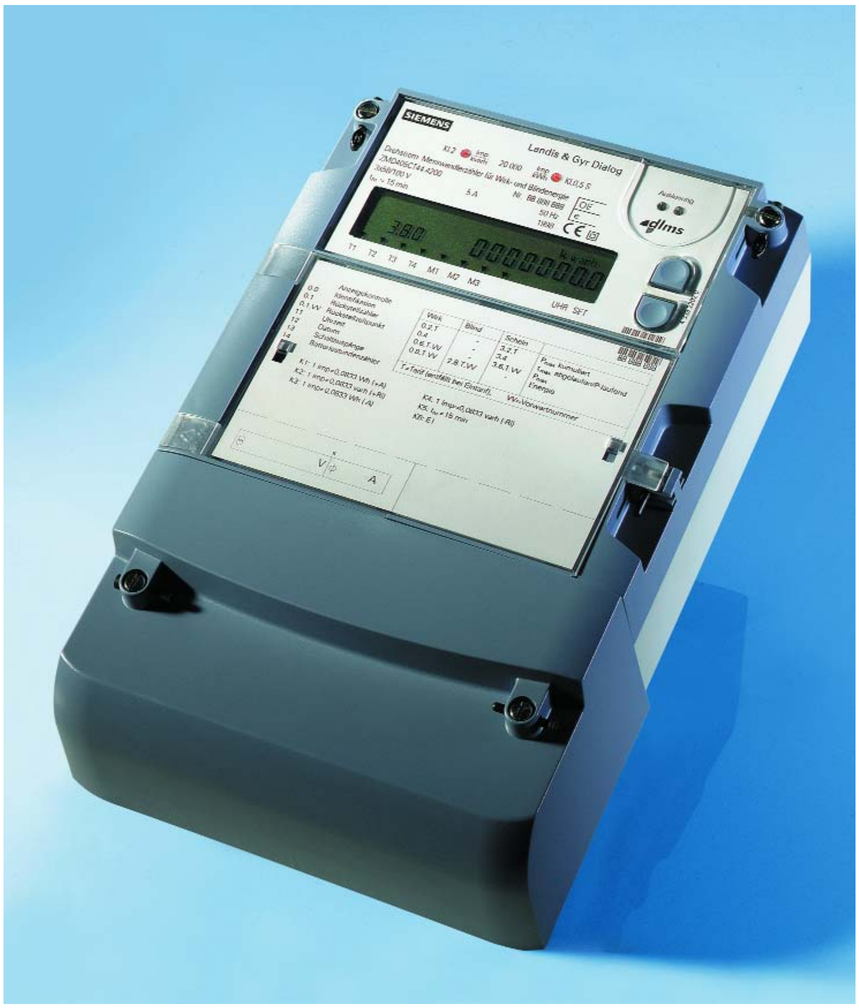


Figure 2. The 'Dialog' model electricity meter manufactured by Siemens featuring an IEC 1107 optical interface.

these features are of course protected by several layers of stringent security checks/passwords and are not available to the consumer.

The meter also stores power consumed and the average power consumption or 'load profile' can be output periodically by using a commercial meter reading hand terminal. These devices are expensive and it would not be possible to justify the purchase of one for home use where it may only be used occasionally or just to satisfy your natural curiosity by exploring the possibilities of this interface.

The simple interface design presented here offers a perfect low-cost alternative. This opto-electronic interface connects an IEC 1107 compliant meter directly to any PC or laptop via the RS232 serial interface port. From a hardware standpoint the design could hardly be simpler even power for the interface is derived from the PC so that no external mains unit or battery is necessary. The accompanying software stores the load profile values in a file so that they can be used in other applications including spreadsheets.

This design offers a simple method for the consumer to monitor actual power consumption using the IEC 1107 interface.

Hardware

Figure 1 shows the complete interface circuit diagram. Power for the circuit is actually derived from the RS232 interface of the PC using signals TXD, DTR and CTS. Operational amplifier IC1 is configured as a comparator with its switching threshold voltage set by preset P1. The potential divider chain formed by R1 and R2 sets the voltage at the inverting input to IC1. Infrared receiving diode D7 is connected in parallel to R1 so that IR light falling on D7 will effectively reduce the resistance of the R1/D7 pair and so produce a waveform at the input to IC1 which corresponds to the received IR signal. The signal is converted to RS232 voltage levels by IC1 so that the output signal can be used to drive the received data line (RXD) directly. Signals to the meter are produced by the infrared emitting diode D8 while R4 limits the forward conduction current.

The data rate defined in the IEC 1107 protocol is so low that a standard 741 type op-amp is also suitable for use as a comparator in this application.

The layout of the circuit is so simple and uncritical that even without a purpose-made PCB construction should present few problems. The only points to look out for are firstly to ensure that you use a 9-way D-type **socket** for connector K1 and not a plug and also that the receiving diode D7 is fitted with a hood or tube so that it can only detect light coming from directly in front of it. If you choose to mount D7 any distance from the rest of the interface circuitry then the cabling to D7 should be screened.

Circuit alignment is necessary to set up the switching threshold of IC1. With the interface connected to a running PC, receiving diode D7 should first be disconnected from the circuit or covered up so that no IR light can reach it. Adjust P1 until the output of IC1 goes low now back-up the preset until just **before** the output of IC1 flips high and this will be the optimum setting of P1.

Assembling the bits

To ensure a good optical coupling between the transducers in the meter and D7 and D8 in the interface probe it is necessary for them to be in close physical contact. Commercial probes are usually fitted with a magnetic collar that attaches to the circular steel washer on the front of the meter. The meter transducers are mounted within the central area of this washer (see **Figure 2**). For our purposes it is simpler to mount D7 and D8 on a small strip of perforated board or plate and fix this to the front of the meter with tape. For the probe itself you will need a suitable piece of perforated strip or plate and using a drill, make two 5 mm holes at a spacing of 6.5 mm about the centre line. Diode D7 is fitted in the left hole and D8 in the right hole. This home-made probe can now be fitted over the reading zone on the meter and fixed with tape or better still Velcro to allow simpler re-attachment.

Software

The software accompanying this project allows the actual load profile to be sent out periodically from the meter. In some countries, the meters output this information every fifteen minutes but in the UK, it is every thirty minutes. The output values are positive and negative, active and reactive power components of the energy consumed. The corresponding documents detailing both the hardware and communication protocols can be found in references [1] and [2] respectively. The measured values should be available, provided the meter stores them. Unfortunately, the presence of a physical IEC 1107 interface on the meter does not guarantee that the load profile is stored in the meter! A call to the technical department of your utility supplier should make the picture clearer. Failing that, the complete interface circuitry is so simple that it can be constructed in an evening and together with the software (available freely from the *Elektor Electronics* website) it should be possible to ascertain relatively quickly if the meter wants to play ball. You will not have wasted much time or expense if the answer is negative and you will certainly gain a better understanding of the communication protocols involved.

Before attempting to run the project software, connect the interface board to the first serial port (COM1) of your PC, preferably before the PC has been switched on. If you intend to use the COM2 port then use executable program with '2' behind its name. The load profile can be stored to any file in ASCII format.

Table 1 shows a load profile taken from a meter made by EMH. The first line gives the date and time that the reading was taken (00-07-11,08:30:00) which you probably have already guessed was the 11th July 2000 at 08:30. The next line shows the measurement units together with the channel number. The third line shows the actual measurements at 08:30 and each successive line shows readings at 15-minute intervals. The positive value of the consumed active power of channel 1 is listed in the first column (02.24). The 5 after the channel number indicates that this is an average reading. The units are in Kilowatts (kW). The average value of the reactive component in channel 3 is shown in the second column (00.28). The average negative active power (power flowing from the consumer into the grid) is shown in the third column (00.00) and similarly the negative reactive component is shown in the fourth column (00.00). Reactive power is expressed in Kilovar (kVar).

The software comprising the executable programs along with its source file is freely available from the *Elektor Electronics* website at: http://www.elektor-electronics.co.uk Click on 'Download' and select number **000195-11** to receive the zipped files containing the executable (.exe) programs and the Pascal Unit (.tpu).

(000195-1)

VA, Var and Kilovar

Few electrical loads are purely resistive even the humble light bulb has a degree of inductance. Non-resistive properties of loads have the effect of introducing a phase shift between the ac voltage applied and the current consumed by the load. The time difference between the voltage and current waveform is expressed as a fraction of the frequency of the ac voltage in degrees and is known as the phase shift. Every waveform with a phase shift between the voltage and current can be represented by two components the **active** current which is in phase with the voltage and the **reactive** current which is shifted by 90° with respect to the voltage. Capacitive loads cause the current to lead the voltage while inductive loads (motors, transformers etc) cause the current to lag the voltage. A simple ammeter will not be able to separate the two values of current; it will instead measure the total current and this will be the geometric sum of the two components. This is called the **apparent** current. Multiplying these values of current with the voltage will give three different values for power. (see the DIN 40110 standard):

– **Active Power** (Active current ? Voltage) expressed in Watts (**W**).

– **Reactive Power** (Reactive current ? Voltage) expressed in Voltampères reactive (**Var**).

– **Apparent Power** (Apparent current ? Voltage) expressed in Voltampères (**VA**).

VA and Var are nothing more than special names for Watts and are used so that we can distinguish between the active, reactive and apparent power. One Kilovar (kVar) is equal to 1000 Var – just as in Kilowatt (kW) and Kilovoltampère (kVA).

Table 1.

Load profile output information (example).

```
P.P(00-07-11,08:30:00)(C@@@@@@@)(15)(14563)
(1.5_kW,3.5_kvar,2.5_kW,4.5_kvar)
(02.24)(00.28)(00.00)(00.00)
(02.28)(00.36)(00.00)(00.00)
(02.28)(00.32)(00.00)(00.00)
```

Some useful Web addresses

Meter manufacturers:
www.siemet.com (select 'metering')
www.emh.de
www.enermet.de
www.dzg2.de
www.abb.de/messtechnik
www.iskraemeco.si

Useful IEC1107 info:
www.abacuselectrics.com/iec1107.htm

Literature:

[1] German Industry standard DIN 43863-3

[2] International Standard IEC 1107/ Norme Internationale CEI 1107

VHF Airband Receiver

a double-conversion superhet for 108-137 MHz NAV and COM reception

Design by G. Baars gert_baars@hetnet.nl

This receiver, specially designed for the VHF airband, couples decent performance to simple construction, all at an affordable price. It does not contain exotic parts and may be adjusted without special instruments, so we reckon the design makes an ideal entry-level receiver for aviation enthusiasts with two feet firmly on the ground.

Eavesdropping on police, ambulance and fire brigade communications, to mention but a few examples, is a hobby with a persistent attraction to many. This has been the case ever since these services started using unprotected mobile communications. The exact reasons for the 'addiction' are hard to pinpoint. Curiosity, of course, but there's more to it. A possible enticing factor is that scanner listening is somewhere in the twilight zone between 'illegal' and 'allowed', which no doubt adds to the excitement enjoyed by many scanner enthusiasts.

One of the most popular frequency ranges to listen to is known as the VHF airband. There, virtually all communications are heard between air traffic controllers, pilots and engineers. The band allows the above mentioned excitement to be coupled to the interest in 'all things aeronautic', and the result is sure to appeal to many.

The VHF airband is generally defined as the frequency range between 108 MHz and 137 MHz, which indicates that it is intended to form a seamless link with the VHF FM broadcast band, 87.5 MHz to

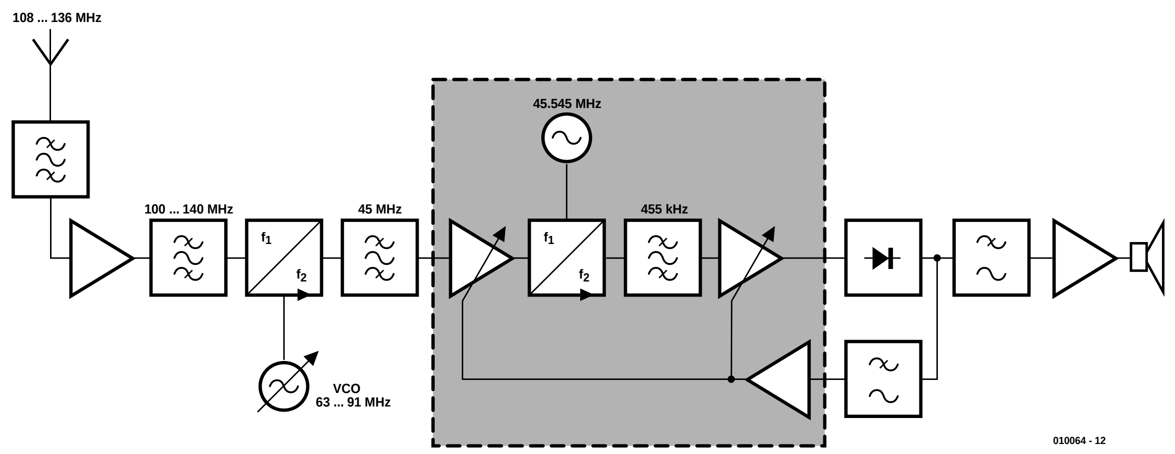


Figure 1. The receiver is a double-conversion superheterodyne design with intermediate frequencies at 45 MHz and 455 kHz.

108 MHz. This could lead us to assume (or hope) that by clever modification, an existing FM broadcast receiver can be 'tweaked' into operation at the low end of the VHF airband. Alas, this is not as easy at it seems at first blush. Firstly, the bandwidth used in the FM broadcast band is much larger than that in the VHF airband, and the same goes for the channel spacing (100 kHz as opposed to 25 kHz). The upshot is that the selectivity of the FM radio will be grossly inadequate. Secondly, all VHF airband communication is firmly regulated to employ amplitude modulation (AM), which would require the existing FM demodulator to be removed and replaced by an AM equivalent. To cut a long story short: let's forget about the FM radio and go for a dedicated VHF airband receiver.

Considerations

To make clear what sort of receiver we'll be discussing next, a short list of important features may be in order. Nearly all issues mentioned below are discussed in greater detail further on in the article as we delve into the electronics.

- Perhaps the most essential feature, the present receiver is a **double-conversion superheterodyne** design, comprising two mixers, two local oscillators (LO) and two intermediate frequency (IF) amplifiers. The superhet principle is sure to result in good receiver performance in respect of image rejection and selectivity.
- The first LO is a VCO (voltage controlled oscillator) with varicap tuning, **fine** and **coarse**.
- Because the project employs **off-the-shelf inductors**, successful construction is not limited to RF specialists like radio amateurs. Only one inductor has to be wound at home — a simple air-cored coil.
- Adjustment does **not require any specialized equipment** and can be done **by listening only**.
- Because the complete receiver including audio amplifier and power supply regulator is accommodated on a **single PCB**, wiring is down to a minimum.
- The **receiver bandwidth** is easy to select by fitting a ceramic filter with a bandwidth of 6 kHz or 15 kHz.
- The receiver has provision for extension by a counter for frequency readout and an external PLL for tuning. Note that we have no firm plans to realize these extensions.

Block diagram

The overall structure of the receiver is illustrated in **Figure 1**.

The RF signal picked up by the whip antenna (length approx. 60 cm) is first filtered to suppress out of band components. Then follows a 20 dB amplifier and a filter with a pass-band of about 100-140 MHz. The main function of this filter is to keep signals at the image frequencies away from the RF amplifier input.

In the first mixer, the amplified and filtered antenna signal is mixed with the output signal of a VCO (voltage controlled oscillator). The VCO has a frequency range of 63 MHz to 91 MHz, and is used to tune the receiver. The difference signal that occurs as a result of mixing the RF and VCO signals has a fixed frequency of 45 MHz. This is called the first IF. Using a 45-MHz filter, the first IF signal is freed from any spurious components.

The first IF signal is then amplified before being applied to the second mixer, where it is heterodyned with a 44.545 MHz signal from a fixed oscillator. The resulting difference signal at 455 kHz is filtered again and then amplified. Next comes the AM demodulator. The bandwidth of the 455-kHz filter determines the overall receiver selectivity.

Behind the demodulator, a signal is shown to pass through a buffer before being applied to the gain stages before and after the second mixer. This is the automatic gain control (AGC) system, which serves to reduce the overall receiver gain when extremely strong signals are received. The AGC levels out large signal strength variations and so prevents you having to re-adjust the volume every time you tune to another signal.

As indicated by the dashed outline in the block diagram, the second mixer, the 44.545-MHz oscillator, the two adjustable-gain amplifiers and the AGC are contained in a single integrated circuit. No doubt this will help to make the construction of the receiver much easier than with discrete components.

Behind the AM demodulator, we find a simple low-pass filter followed by a small audio power amplifier and of course the usual loudspeaker.

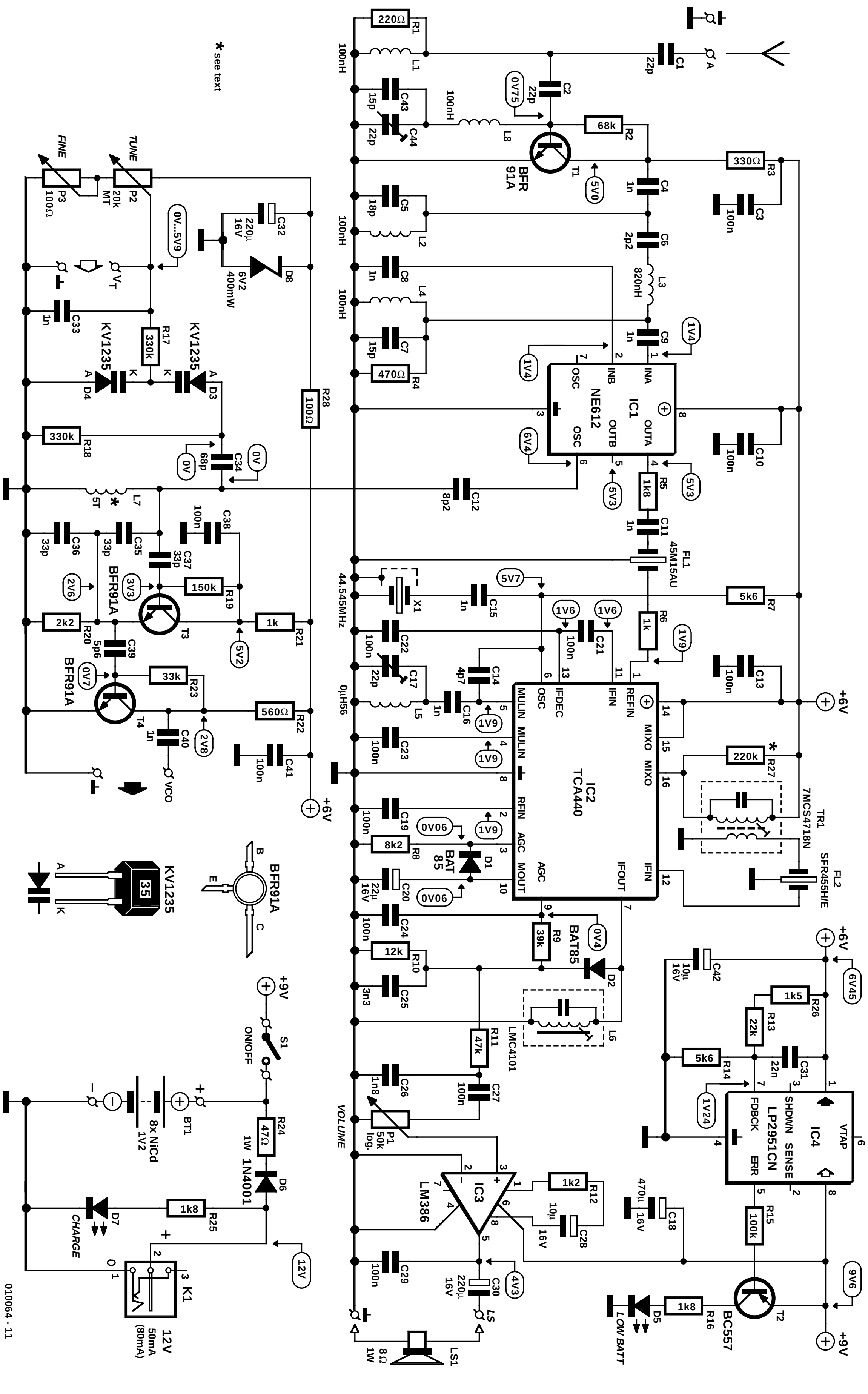


Figure 2. Thanks to the use of an integrated mixer/oscillator/IF amplifier chip type TCA440, the circuit diagram is relatively uncluttered.

Practical realisation

The circuit diagram of the VHF Airband Receiver is given in **Figure 2**. Let's have a look how the functions discussed above get their practical realisation.

The antenna signal arrives on L1, with a notch consisting of L8-C43-C44 added for suppression of unwanted signals. The RF input amplifier, T1, is a type BFR91 bipolar transistor. This device ensures a fair amount of gain at an acceptable noise figure. The 100-140 MHz bandpass is a 3-pole Butterworth filter consisting of L2-L3-L4-C5-C6-C7. This network, helped by the 'coarse' filter at the input, provides about 50 dB worth of image rejection.

The first mixer employs the well-known NE612 IC, which receives the VCO output signal at its pin 6 via coupling capacitor C12.

The VCO is built around transistor T3, another BFR91 in a modified Colpitts configuration which is a classic circuit in RF technology and known for its good stability. The oscillator's resonant circuit is tuned by two variable-capacitance ('varicap') diodes, D3 and D4, whose capacitance is an (non-linear) inverse function of the tuning voltage applied across them via their common cathode. The tuning voltage may be adjusted between 0.5 V and about 6 V using potentiometers P2 (coarse) and P3 (fine). Network R28-D8 acts as an extra stabilizer on the tuning voltage, and helps to counteract VCO frequency drift causing detuning of the receiver.

Via connection V_T, the varicap control voltage is made externally accessible in case it is decided (at a later stage) to use a PLL synthesizer to tune the receiver.

Along the same lines, the VCO output signal is made available via buffer T4 to allow a frequency readout to be connected. If you do not plan to use such an extension, you may safely omit T4, C39, R22 and R23 when building up the circuit on the PCB.

The filter at the output of the first mixer is a 45-MHz ceramic type with a nominal bandwidth of 15 kHz. The filter is followed by the section in the dashed outline shown in the block diagram. All of these functions (preamplifier, mixer, oscillator, IF amplifier and AGC) are contained in the TCA440 integrated circuit, which (almost) forms a single-chip radio receiver. Of course, some external components are needed for the job. Of the more or less standard components around the TCA440 (mostly resistors and components), the most important are without doubt the 44.454-MHz crystal, X1, LC combination L5-C17 for the internal oscillator and the 455-kHz bandpass filter consisting of transformer Tr1 and ceramic filter FL2. Inductor L6 acts as an output tuned circuit.

Further towards the output of the circuit we find a simple diode detector, D2, for AM demodulation, a low-pass filter R10-R11-C25-C26 and, finally, an integrated audio amplifier type LM386, IC3.

Power supply

The receiver was designed to operate from an unstabilized 9 V supply voltage. The supply voltage directly powers audio amplifier IC3, as well as voltage regulator IC4, which supplies a stabilized 6-V rail (actually, 6.45 V) for the rest of the receiver circuitry. Because the 'error' output of IC4 (pin 5) goes low when the input voltage drops between the minimum level for proper stabilisation, it is used to control a 'LowBatt' indicator LED via transistor T2. The minimum voltage drop across IC4 being a mere 0.1 V, the battery can be 'juiced' before LED D5 will light to indicate that it's definitely flat.

The receiver draws about 60 mA with a loudspeaker connected, and about 35 mA if you use 32-Ω headphones with both earpieces connected in parallel. Consequently, a 9-V PP3 battery will last for about 5 or 10 hours, respectively. If you need more battery capacity, you may consider using eight 1.2-V NiCd penlight-size batteries (AA), as indicated in the circuit diagram. These batteries may be charged by connecting a 12-V mains adaptor to K1. LED D7 then acts as a charging indicator, while resistor R24 determines the level of the charging current. The indicated value of 47 Ω results in a (generally safe) charging current of about 50 mA. This allows the mains adaptor to remain on and connected up without problems, irrespective of the exact type of battery used.

If the receiver is used with non-rechargeable batteries only, components R24, D6, R25, D7 and K1 may be omitted to reduce cost.

Tuning and selectivity

As already mentioned, ceramic filter FL2 determines the selectivity of the receiver. Two options are available: a filter with a bandwidth of 6 kHz (SFR455H or the CFW455H), or 15 kHz (SFR455E or CFW455E). Although you may want to go for the highest selectivity straight away, we would advise using the 15-kHz version, at least to begin with. Radio equipment that conforms to the 8.33-kHz channel spacing standard (introduced in 1999 for ATC communications) is still a bit thin on the ground, 25 kHz still being the most widely used channel distance. Also, tuning the receiver is much more difficult when using a 6-kHz filter. Despite the use of a multiturn pot for P2, you would easily miss stations. Of course there's the fine tuning control P3 but this is of little use once you've tuned past the signal already.

However, if an external PLL synthesizer is used to tune the receiver, it is better to go for the narrower filter if only because it reduces the noise level.

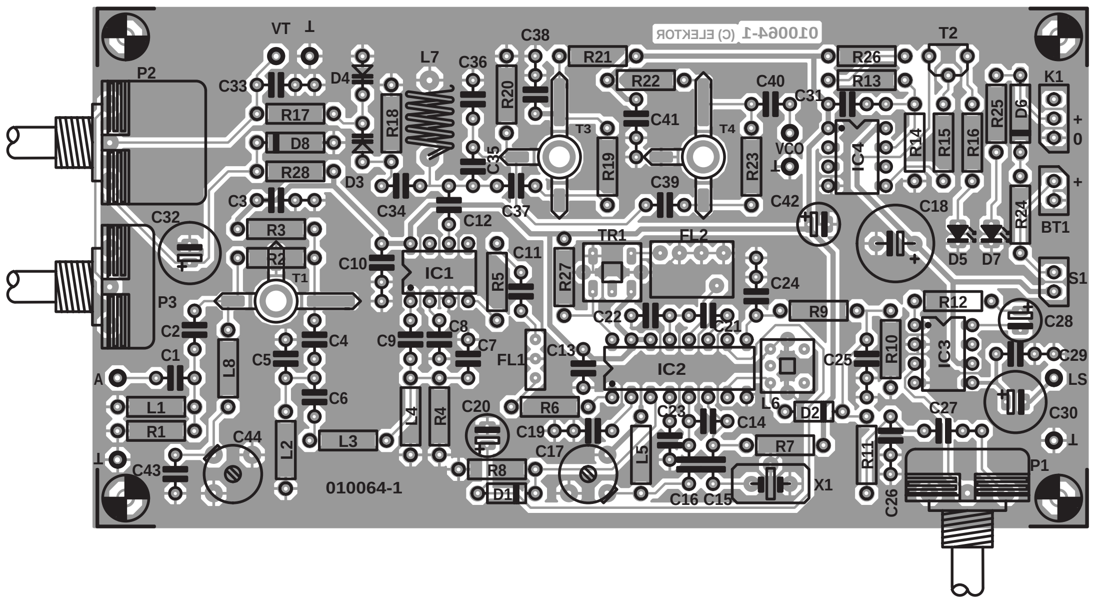
A final note regarding the tuning — some drift may be noted immediately after the receiver is switched on. The effect should disappear after a 5-minute warm up period.

Construction

Figure 3 shows the copper track layout and component mounting plan of the printed circuit board we've designed for the receiver. The board actually accommodates the circuit shown in Figure 2, that is, including audio amplifier IC3, regulator IC4 and the NiCd charger circuit consisting of R24, R25, D6, D7 and K1.

Despite a fair number of components on the board, construction is mostly plain sailing. As usual, make sure you fit the polarized components the right way around — we mean integrated circuits (look for the notch), electrolytic capacitors, transistors and diodes. Varicap diodes D3 and D4 require particular attention because they do not have a clear marking. If you hold the diode such that the type code is legible with the pins downwards, then the left leg is the anode, and the right leg, the cathode. On the board, D3 and D4 are not fitted in the same direction, so watch out!

Construction is best started by fitting the low-profile components simply because that is most convenient. So, start with the resistors, then the smaller capacitors, the electrolytics, and so on. Sockets may be



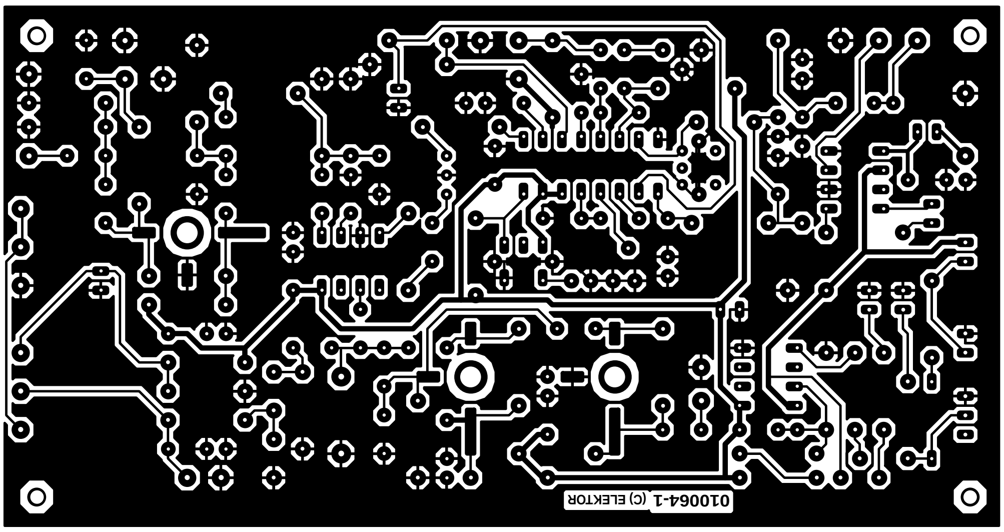
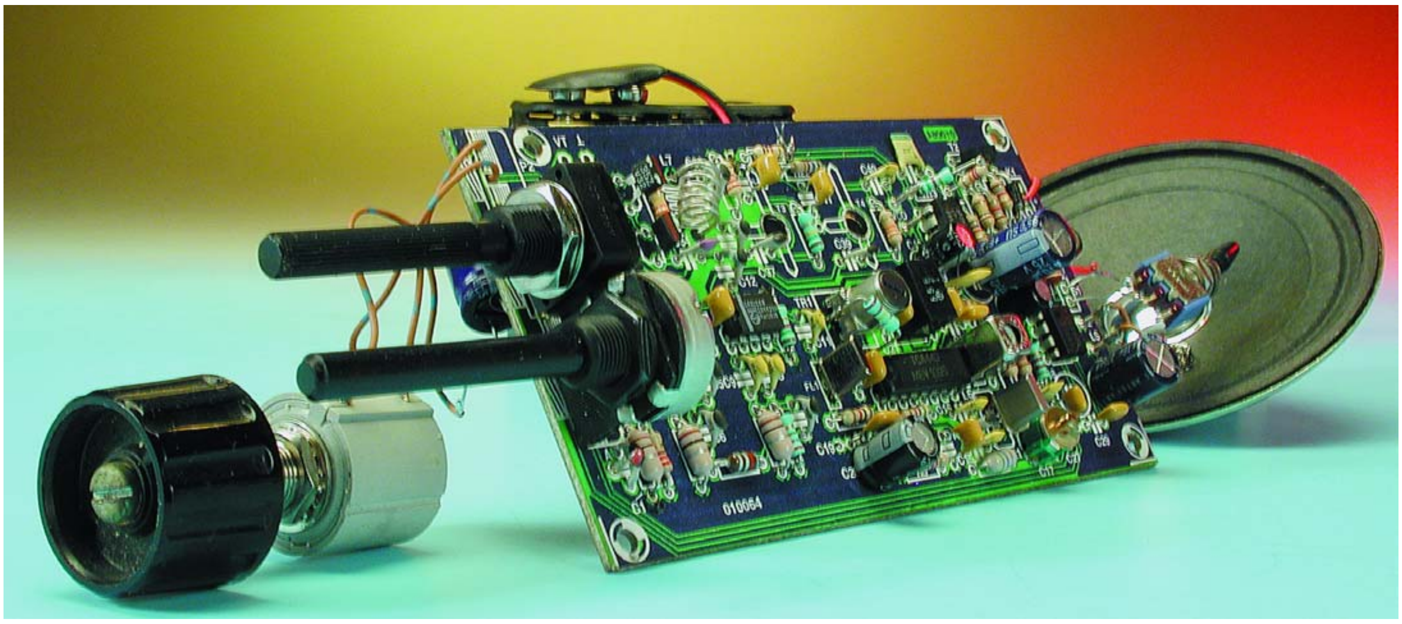


Figure 3. Copper track layout and component mounting plan of the PCB designed for the receiver (board available ready-made).



used for IC3 and IC4, while IC1 and IC2 may be soldered directly on to the board.

RF transistors T1, T3 and T4 (only if a buffered VCO output is required) are soldered at the bottom side of the board, with their legs directly onto the relevant copper tracks. They will only fit in one way and holes are provided in the PCBs for their round cases to be seated in.

Next, the inductors. L1-L5 and L8 are ready-made miniature chokes that look like precision resistors, complete with coloured bands indicating the value. IF transformer Tr1 and tuned circuit L6 are also off-the shelf components. Both are housed in metal cases that will only fit one in one way. The only inductor to be wound at home is L7. Easy, really, because it consists of 5 turns of silver-plated wire with a diameter of 1 mm. The inside diameter is 5 mm obtained from a drill bit or a pencil. After winding the inductor, space the windings evenly by pulling them apart until an overall length of about 12 mm is obtained.

A few more details about populating the board. Resistor R24 should be a 1-watt type, mounted slightly above the board because it may get a little warm. Resistor R27 is not used because our testing of the receiver indicated that it was not required. Indicator LEDs D5 and D7 have to be mounted so that they can be seen from the outside. In most cases, that will require connecting them to the board via light duty flexible wires. The metal case of quartz crystal X1 has to be (quickly) soldered to ground using a very short piece of leftover component wire.

You will find that potentiometers P1, P2 and P3 will fit the board directly. However, whether or not that is actually done depends mostly on the enclosure you have in mind for the receiver. P2 and P3 may be connected to the board using flexible wire. P1 on the other hand will require a short piece of screened audio cable.

Having fitted all the components on the board, it is a good idea to use a multimeter to check the indicated measurement point for the correct voltages. If they are (roughly) correct, you may safely assume that there are no constructional errors in the circuit.

As a further aid in getting the project to work without too much time spent on fault-finding, **Figure 4** shows the wiring diagram of the complete receiver, with the PCB at the centre of things.

Mechanical work

Having modest dimensions, the printed circuit board should fit in a reasonably compact case, together with the receiver's loudspeaker and the batteries. Although we have no grave objections against a plastic (ABS) enclosure, a metal one is highly recommended because it minimizes the risk of VCO detuning by the so-called 'hand effect'. Our prototype of the VHF Airband Receiver was built into an aluminium diecast enclosure type BIM5005 which has outside dimensions of 15?8?5 cm. Although the board will fit neatly into this case, we should add that space is at a premium if the NiCd batteries and the loudspeaker have to be squeezed in as well. For example, near multiturn pot P2 we had to remove some aluminium from the inside of the lid.

Figure 5 allows an inside view of the prototype receiver. The antenna we used for our experiments was a common or garden telescopic rod.

COMPONENTS LIST

Resistors:
R1 = 220Ω
R2 = 68kΩ
R3 = 330Ω
R4 = 470Ω
R5,R16,R25 = 1kΩ8
R6,R21 = 1kΩ
R7,R14 = 5kΩ6
R8 = 8kΩ2
R9 = 39kΩ
R10 = 12kΩ
R11 = 47kΩ
R12 = 1kΩ2
R13 = 22kΩ
R15 = 100kΩ
R17,R18 = 330kΩ
R19 = 150kΩ
R20 = 2kΩ2
R22 = 560Ω
R23 = 33kΩ
R24 = 47Ω (1W)
R26 = 1kΩ5
R27 = not fitted
R28 = 100Ω
P1 = 50kΩ logarithmic potentiometer
P2 = 20kΩ multiturn
P3 = 100Ω linear potentiometer.

Capacitors:
C1,C2 = 22pF
C3,C10,C13,C19,C21-C24,C27,C29,C38,C41 = 100nF
C4,C8,C9,C11,C15,C16,C33,C40 = 1nF
C5 = 18pF
C6 = 2pF2
C7,C43 = 15pF
C12 = 8pF2
C14 = 4pF7
C17,C44 = 22pF adjustable (trimmer)
C18 = 470µF 16V radial
C20 = 22µF 16V radial
C25 = 3nF3
C26 = 1nF8
C28,C42 = 10µF 16V radial
C30, C32 = 220µF 16V radial
C31 = 22nF
C34 = 68pF
C35,C36,C37 = 33pF
C39 = 5pF6

Semiconductors:
D1,D2 = BAT85
D3,D4 = KV1235
D5 = LED, red, high efficiency
D6 = 1N4001
D7 = LED, green, high efficiency
D8 = zener diode 6.2V, 0.4W
IC1 = SA612AN or NE612
IC2 = TCA440
IC3 = LM386
IC4 = LP2951CN
T1,T3,T4 = BFR91A
T2 = BC557

Miscellaneous:
BT1 = 9V battery (PP3) or 8 NiCd batteries (1.2V)
FL1 = 45M15AU
FL2 = SFR455H or -E (CFW455H or -E)
K1 = mains adaptor socket, PCB mount
L1,L2,L4,L8 = 100nH
L3 = 820nH
L5 = 560nH
L6 = LMC4101 (Toko)
L7 = 5 turns ∆1mm silver-plated wire on ∆5mm former (no core)
S1 = switch, 1 make contact
TR1 = 7MCS4718N (Toko)
X1 = 44.545MHz quartz crystal (case connected to ground)
LS1 = loudspeaker 8Ω 1W
PCB, order code **010064-1** (see Readers Services pages)
Enclosure: e.g., BIM, dim. 150×80×50mm, order code 06.11.5005 (normal) of 06.11.5105 (enamel finish)

Many RF parts for this projects, including inductors, varicaps ceramic filters and trimmers are available from Barend Hendriksen HF Elektronica BV, PO Box 66, NL-6970-AB, Brummen, The Netherlands. Tel. (+31) 575 561866, Fax (+31) 575 565012. Website www.xs4all.nl/~barendh/, email barendh@xs4all.nl.

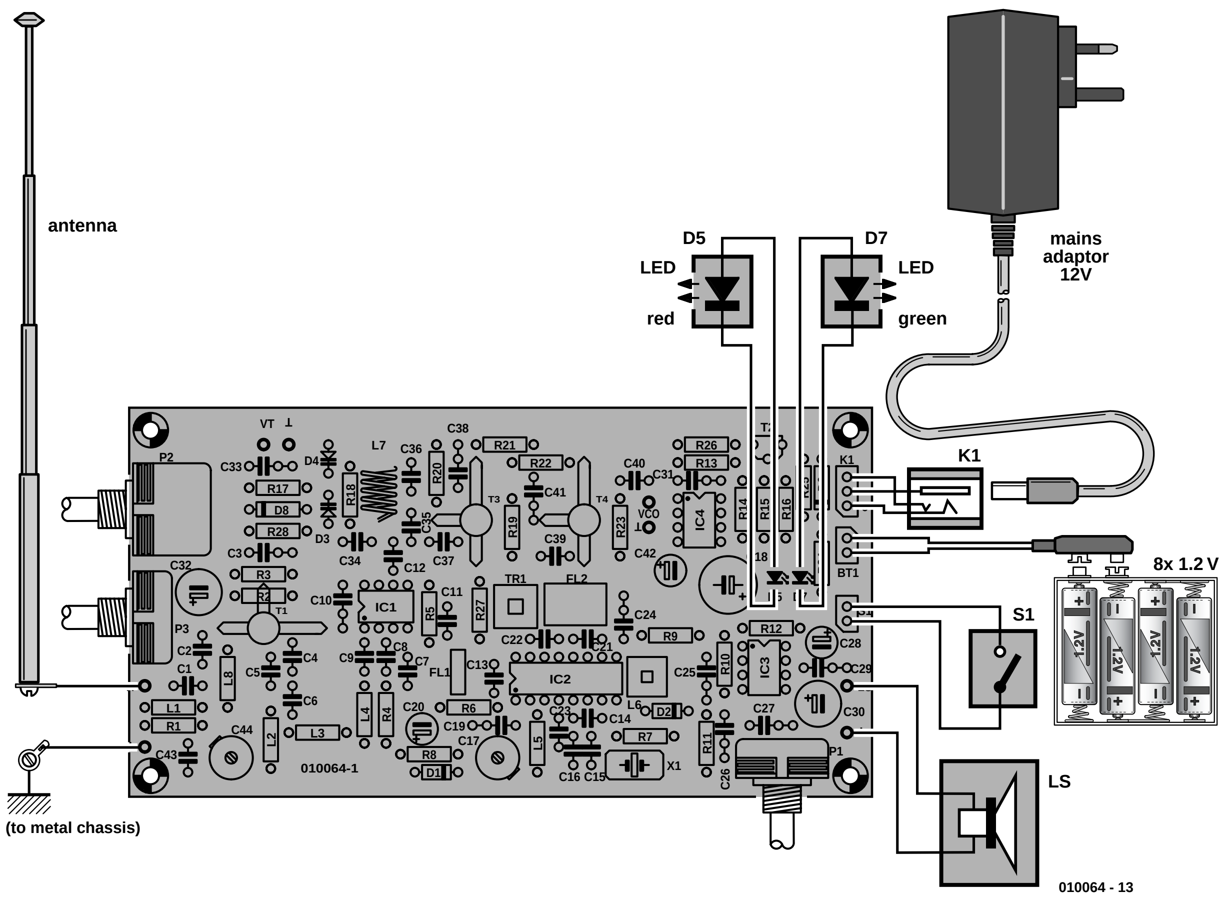


Figure 4. Overview of external controls and other elements connected to the board.

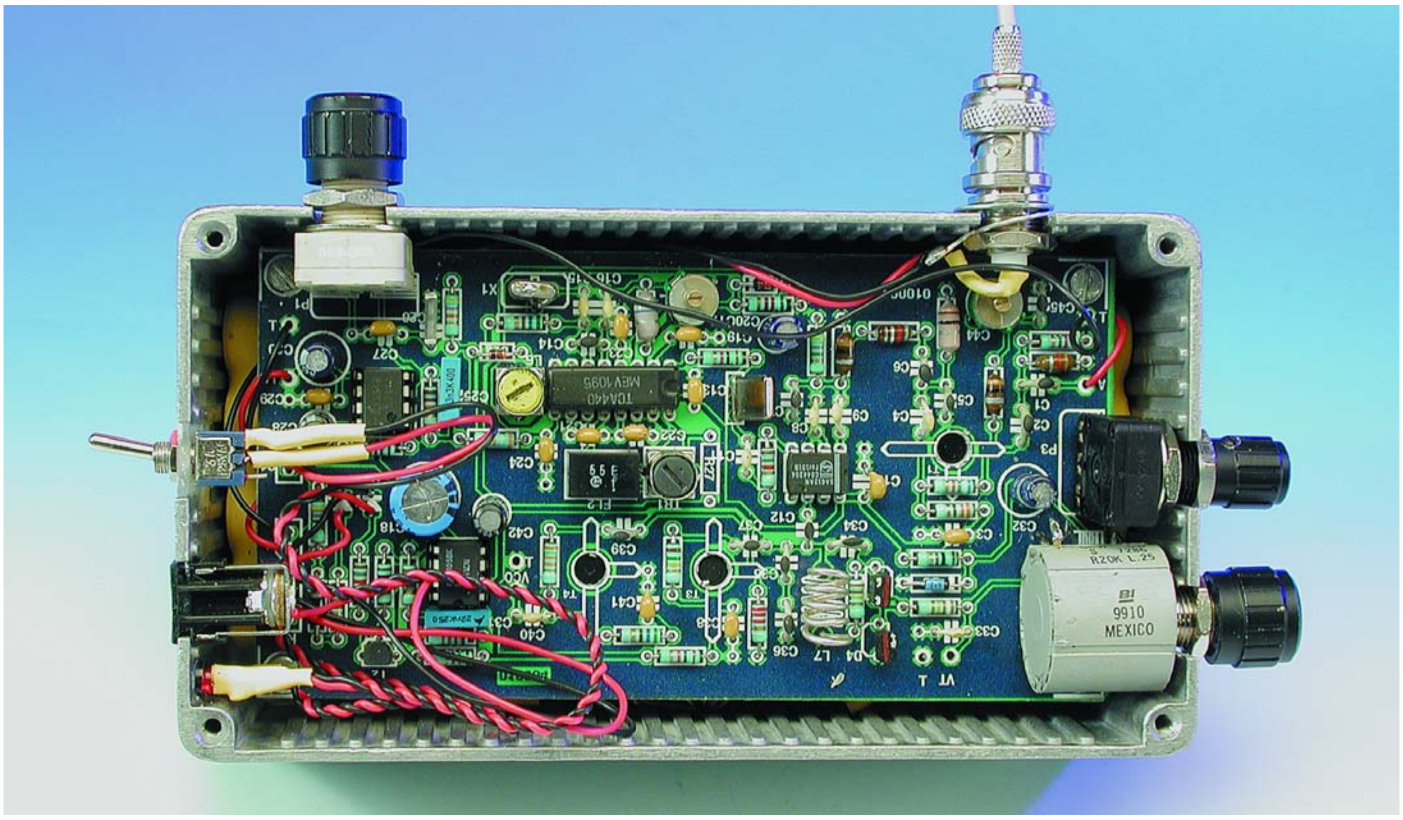


Figure 5. The PCB and ancillaries are a tight fit!

Alternatively, you may want to use a piece of rigid wire with a length of 60 cm or so, mounted in a banana plug.

Adjustment

There are four adjustment points in the receiver. The cores of Tr1 and L6, as well as trimmer C17, are simply adjusted for maximum noise output. Trimmer C44 is set to mid-travel and may be re-adjusted later to cancel breakthrough of strong signals from nearby FM broadcast stations. That's it, really!

If you have closely followed the winding directions for inductor L7, the VCO should be up and running with the correct tuning range, which may be verified if you have a frequency meter available — connect it to the VCO output and turn P2 to see if the VCO can be tuned between 63 and 91 MHz. If necessary, tweak the tuning range by compressing the turns of L7, or pulling them further apart. Make small adjustments at a time!

Reception

Most air-traffic communication may be picked up in the so-called COM (communications) section of the band, between 117 and 137 MHz, The lower part, 108-117 MHz, is reserved for beacons, in-flight landing systems (ILS), navigation beacons and other utility systems, hence it is often referred to as NAV. The best way to find out about the frequencies used on or near the airport you live close to, is to consult a Scanner Guide, which are available in several countries.

Using the HP8640B signal available in the *Elektor Electronics* design laboratory, the sensitivity of the receiver was measured at about 0.5 µV for 12 dB (S+N/N). This should be sufficient to pick up communication between air traffic controllers and pilots at a distance of more than 25 kilometres from any major airport. At first, you may be surprised to note that the aircraft signal is often stronger than that of the control tower, but bear in mind that the aircraft is up in the sky so its reception path will have a minimum of obstacles!

Finally, by tuning the receiver to weak navigation beacon signals, it can be used as an excellent propagation monitor to predict sporadic-E openings in the VHF band.

(010064-1)

Image rejection

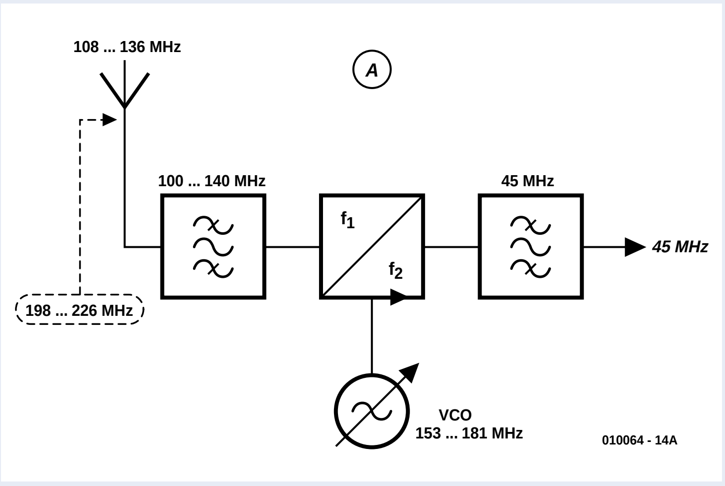
Inherent to its design, any superheterodyne receiver (single or double conversion) is in principle open to two bands, the desired band and the 'image frequency' band. These bands are spaced apart two times the first intermediate frequency. Image frequencies are caused by unwanted output products of the mixer(s) used.

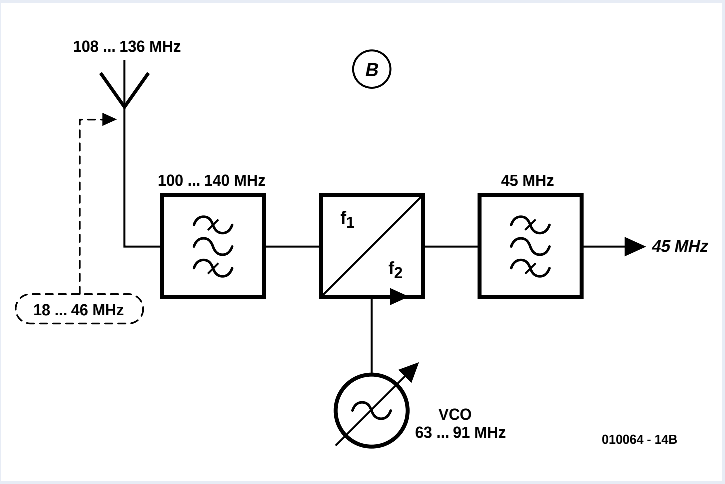
In a superheterodyne receiver, the received signal (RF) is mixed with a local oscillator (LO) signal, in such a way that the mixer output produces an intermediate frequency (IF) which is constant over the entire frequency range. In the receiver shown in **Figure A**, the RF signals are in the desired band between 108 MHz and 136 MHz, and the LO signal can be tuned between 153 MHz and 181 MHz. This is called **high-side injection**. The difference frequency is simply LO–RF = 45 MHz being the centre frequency of the IF passband.

However, from simple mathematics it follows that an identical 45 MHz signal is produced by RF signals between 198 MHz and 226 MHz, as indicated in dashed type. The filter fitted ahead of the mixer has a passband that corresponds to the desired frequency range, i.e., 100-140 MHz, and so serves to suppress signals picked up in the 'image band'.

In this case, at an intermediate frequency of 45 MHz, the image band is less than an octave away from the desired band. Consequently, the passband filter needs to have pretty steep skirts. Alternatively, its tuning needs to 'track' the VCO. Both solutions are relatively difficult to implement, which is not what we are after.

In this example the best way to achieve good image rejection is to resort to **low-side injection**. After all, using a VCO tuning range of 63-91 MHz again results in a fixed IF of 45 MHz (RF–LO). As shown in **Figure B**, the image band is then between 18 MHz and 46 MHz, which is — on average — 2 octaves away from the input filter passband. As a result, these image frequencies can be adequately suppressed using a relatively simple passband filter.



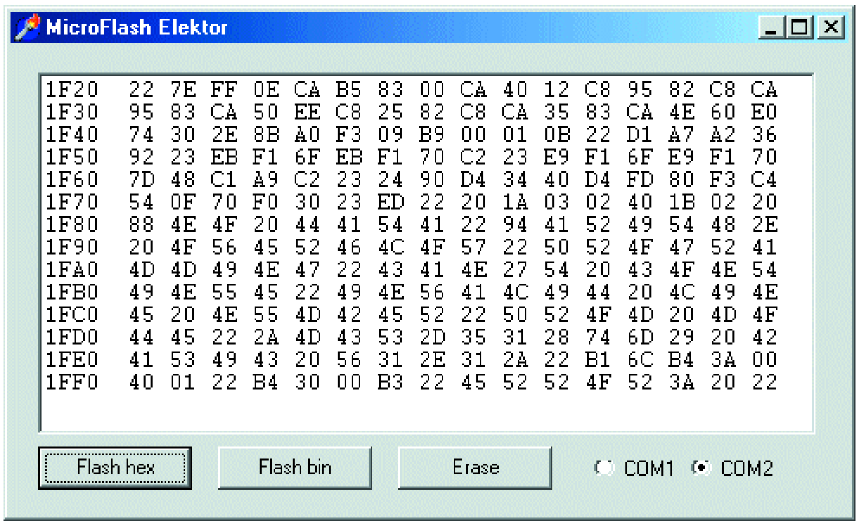


Microcontroller Basics Course (3)

part 3: BASIC-52

By B. Kainka

In the first two instalments of the course, we worked with assembler, but now it's time to use a high-level language: BASIC-52.



```
1F20   22 7E FF 0E CA B5 83 00 CA 40 12 C8 95 82 C8 CA
1F30   95 83 CA 50 EE C8 25 82 C8 CA 35 83 CA 4E 60 E0
1F40   74 30 2E 8B A0 F3 09 B9 00 01 0B 22 D1 A7 A2 36
1F50   92 23 EB F1 6F EB F1 70 C2 23 E9 F1 6F E9 F1 70
1F60   7D 48 C1 A9 C2 23 24 90 D4 34 40 D4 FD 80 F3 C4
1F70   54 0F 70 F0 30 23 ED 22 20 1A 03 02 40 1B 02 20
1F80   88 4E 4F 20 44 41 54 41 22 94 41 52 49 54 48 2E
1F90   20 4F 56 45 52 46 4C 4F 57 22 50 52 4F 47 52 41
1FA0   4D 4D 49 4E 47 22 43 41 4E 27 54 20 43 4F 4E 54
1FB0   49 4E 55 45 22 49 4E 56 41 4C 49 44 20 4C 49 4E
1FC0   45 20 4E 55 4D 42 45 52 22 50 52 4F 4D 20 4D 4F
1FD0   44 45 22 2A 4D 43 53 2D 35 31 28 74 6D 29 20 42
1FE0   41 53 49 43 20 56 31 2E 31 2A 22 B1 6C B4 3A 00
1FF0   40 01 22 B4 30 00 B3 22 45 52 52 4F 52 3A 20 22
```

COM1 [✓ COM2]

Figure 1. Downloading the Basic interpreter.

This interpreter for 8052 microcontrollers is well known to all microcontroller users and universally popular. It is located on the program diskette for the Flash Board in the form of a program file in Intel hex format with the name `BASIC-52.hex`. All you have to do is to use the `MicroFlash.exe` program to load this file into the microcontroller (**Figure 1**). The interpreter is 8 kB long and takes around a minute to transfer.

In order to use the interpreter, you need a terminal emulator program that can send instructions and programs to the microcontroller via the serial interface. Here we use the program `BASIC.EXE` from H.-J. Berndt. When you first start the program, you must configure the interface to be used (COM1: or COM2:). Communications between the microcontroller system and the PC then take place via connector K1. As shown in **Figure 2**, the program has two text windows. The upper editing window is used to edit Basic source texts. Each line can be edited as much as desired; it is not sent to the microcontroller until you press <Return>. The lower text window is a direct terminal. All characters entered using this window are immediately sent to the microcontroller. The responses from the system also appear in this window.

Following a Reset, BASIC-52 first initialises itself and checks the available RAM in the system. After this, the interpreter waits for a 'space' character from the terminal. This is used to automatically determine the baud rate being used and to configure the serial interface of the microcontroller accordingly. This means that you must not press any key other than the 'space' key (ASCII 20h). BASIC-52 will respond with the following start-up message:

```
*MCS-51(tm) BASIC V1.1*
READY
>
```

Now you can enter direct commands or program lines. The query `Print MTOP` returns the highest address of the connected and recognised memory (Memory Top), which in this case is thus '32767' for 32 kB of RAM. When entering known keywords, you do not have to pay attention to upper or lower case. However, the interpreter converts everything internally into upper case, so program listings always appear in upper case. You can try this out using a small test program called Test1.bas (see **Figure 3**):

```
10 for n= 1 to 10
20 print n
30 next n
```

The `LIST` command causes the program to be output via the serial interface and appear in the terminal window. It can be started using `RUN`. Our first program generates an increasing series of numbers in the terminal window.

Commands such as `LIST`, `RUN` and `NEW` can be directly entered as text. However, you can also use the built-in functions in the Program menu. Program/List causes the listing to be copied to the editing window. Once it is in this window, it can be either be modified or stored on the hard disk.

Now it's time for a brief explanation of our sample program. In BASIC-52, each line has a line number (although this is no longer required in modern Basic systems). In line 10, the variable `N` is defined. This is then incremented from 1 to 10 in ten steps. `FOR… TO… NEXT` forms a loop that is executed until the value 10 is reached. This means that `N` receives the value '1' on the first pass through the loop, '2' on the second pass and so on, up to the value '10' on the final pass. In line 20, the current value is output via the serial interface and thereby written to the terminal screen.

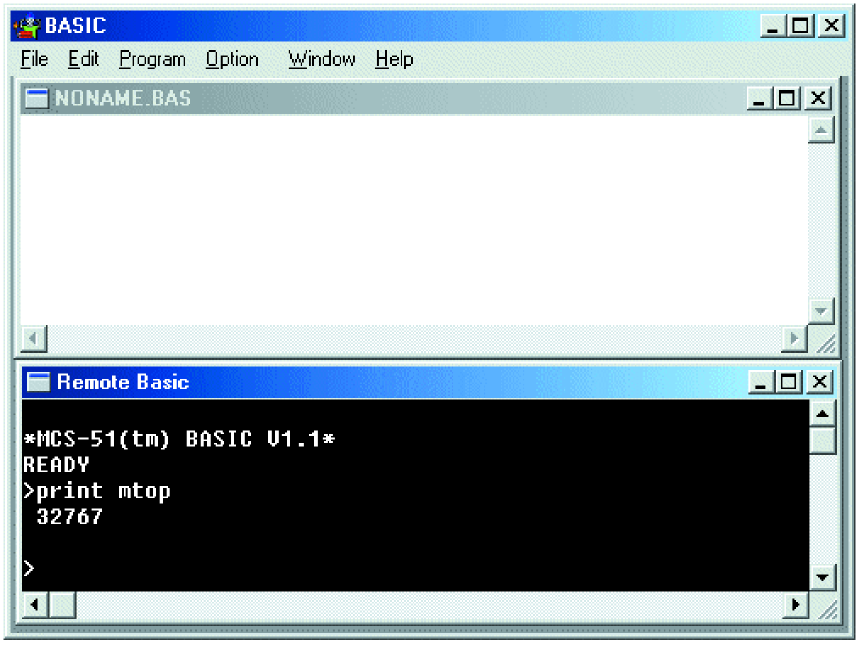
A counting loop can also be built in assembler with a minimum of effort. However, the command `PRINT` is so complex that it would take a relatively elaborate assembler program to achieve the same function. To start with, BASIC-52 uses not only bytes (range 0–255), which can be directly understood by the microcontroller, but also real numbers, each is of which is composed of six bytes representing eight significant digits, a sign and an exponent. The `PRINT` command must convert these numbers into text and send this text character by character via the serial interface. This requires the initialisation of the interface, which is performed by the interpreter when it starts up.

This is not the place for a complete and boring explanation of all of the interpreter's keywords and commands. A brief command summary can be found in the Help file for the editor program. An extensive command summary, complete with a user's manual, can be found in the Free Downloads area of the *Elektor Electronics* website at www.elektor-electronics.co.uk. In the course of the following experiments, you will be introduced to an increasing number of new keywords and learn how to use them. We will start with very simple examples that are functionally fully comparable to the assembler programs that you have already seen. It is particularly interesting to compare the speeds of these functionally equivalent programs. We start with Test2.bas:

```
10 FOR N=0 TO 255
20 PORT1=N
30 NEXT N
40 GOTO 10
```

Here we again use our well-known counting loop, this time with a range of 0–255, which covers the numerical range of a single byte. In BASIC–52, a number can be output to Port 1 just as easily as to the monitor screen. `PORT1=N` transfers the value of the variable `N` to the port. In the opposite direction, the states of all eight port lines can be read using the instruction `N=PORT1`.

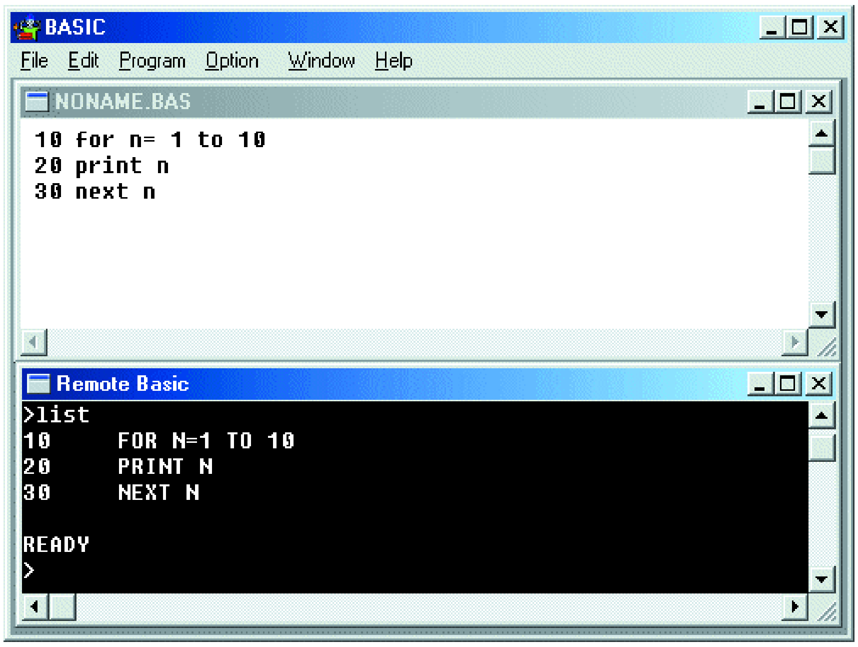
As soon as you start the program



```
*MCS-51(tm) BASIC V1.1*
READY
>print mtop
 32767

>
```

Figure 2. The terminal emulator program used in the course.



```
10 for n= 1 to 10
20 print n
30 next n
```

```
>list
10      FOR N=1 TO 10
20      PRINT N
30      NEXT N

READY
>
```

Figure 3. Entering a new program.

using RUN, you can see how fast it is. A square-wave signal with a frequency of around 200 Hz will appear at P1.0. At P1.7, the signal level will change approximately once per second, so it can be used to directly drive an LED. The comparable assembler program was somewhat faster, even with a supplementary delay loop. Basic programs run between 100 and 1000 times slower than equivalent assembler programs. This is no surprise, since BASIC–52 is an interpreter, which means that it is a rather complicated program that interprets and executes Basic source text word by word. The fact that this is a complicated task can already be seen from the size of the interpreter (8 kB). All of our previous sample assembler programs in the first two instalments of the course, taken together, amount to less than 100 bytes. In these examples, we only wrote very simple, linear programs. The interpreter, by contrast, must perform a very extensive list of tasks. At the run time of a Basic program, the keywords must be recognised and evaluated. Furthermore, the interpreter treats all numerical values as real numbers, which requires significantly more effort than processing bytes.

A frequency divider in Basic

Now it's time to give BASIC–52 a concrete assignment: an input must be constantly monitored for pulses. After exactly 10 pulses, an output toggles. After a total of 20 pulses, the output resumes its original state. The program thus effectively forms a frequency divider with a division factor of 20. Here we are using a program to carry out a task that is usually handled by a few ICs. The interesting question is to determine the limits within which this is possible.

In BASIC–52, there is no simple way to address the individual port bits. Consequently, we always have to read or write an entire port. Information about the state of an individual bit can be obtained by performing an AND operation on the total state of the port (masking). BASIC–52 has the `.AND.` byte operator for this purpose. The two full stops located before and after 'AND' indicate bit processing, in contrast to the logical linking of expressions (`IF` condition 1 `AND` condition 2 ...) In the following listing, this masking is applied in lines 110 and 130. Practically speaking, this command performs eight simultaneous AND operations between two different sets of eight bit states. The value of each resulting bit will be '1' only if both bits in the same position have the value '1'. This can be illustrated by the following examples:

```
10101010 .AND.
00000001 =
──────────────
00000000

11110001 .AND.
00000001 =
──────────────
00000001
```

As you can see, an .AND. operation with 00000001b = 01h = 1 yields only the values '0' and '1'. This means that quite deliberately, only bit 0 of the port is observed, with the behaviour of the remaining bits being masked out. Effectively, we have covered the port with a mask that allows only one bit to be seen, which is the bit that has been chosen as the input. The same thing can also be done in assembler, by the way, although in this case it would not be necessary, since it is possible to directly poll individual bits.

The listing of our frequency divider program (divide.bas) looks like this:

```
100  COUNT=0
110  INP=PORT1.AND.1
120  IF INP=1 THEN GOTO 110
130  INP=PORT1.AND.1
140  IF INP=0 THEN GOTO 130
150  COUNT=COUNT+1 :
       REM PRINT  COUNT
160  IF COUNT=10 THEN
       GOTO 200
170  IF COUNT=20 THEN
       GOTO 250
180  GOTO 110
200  REM Output low
210  PORT1=253:REM P1.1 = 0
220  GOTO 110
250  REM Output high
260  PORT1=255:REM P1.1 = 1
270  COUNT=0
280  GOTO 110
```

Basic programs frequently contain many `GOTO` jumps, which reduces their readability, particularly in the case of large projects. This is often referred to as 'spaghetti code', which means code that is difficult to unravel and chaotic. The frequency divider program is a good example of this. However, it also shows that this style is quite reasonable for relatively small projects. Large programs, on the other hand, unconditionally require a better structure, which can for example be achieved using subroutines.

The program uses one variable (`COUNT`) as a pulse counter and a second variable (`INP`) for the input state of Port 1.0. At the beginning of the program, `COUNT` is set to zero. The program then passes through two loops in

MCS BASIC-52 V1.3

The most recent revision to the original version of the interpreter, which was made by H.-J. Böhling and D. Wulf, has already been presented in the February 2001 issue of *Elektor Electronics*. This version of the interpreter has also been successfully tested with the Flash Board. The program has special commands for programming and erasing EEPROMs. Several different Basic programs can be stored in a single EEPROM and started selectively. The individual steps, as listed below, have been tested using an 8-kB EEPROM:

1. Write a program.
2. Using `XFER`, program an EEPROM with this program.
3. Using `XFER`, load additional programs.
4. `PROG` returns the number of stored programs (e.g. 12).
5. Use `ROM` to switch to the EEPROM.
6. Use `ROM2` to activate the second program.
7. Use `PROG2` to make the first program in the EEPROM an autostart program.

Now whichever program is the first program in the EEPROM will start automatically after a reset, with no need for a terminal.

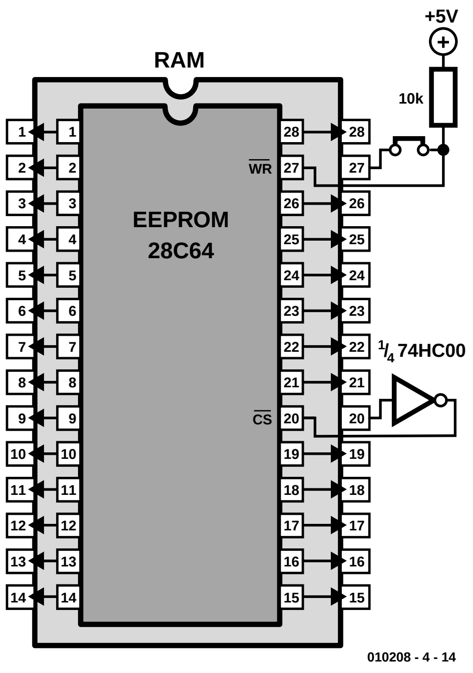


Figure 4. EEPROM wiring.

which the port state is continuously read and evaluated. As long as a High state is seen, the program remains within lines 110 and 120. Lines 130 and 140 handle the Low state in exactly the same manner. The loop is only closed when a new High state again occurs, causing a jump from line 180 back to line 110, where the whole process starts all over again. However, on the way to line 180 the value of the counter is incremented in line 150. Lines 160 and 170 test whether certain count states have been reached (10 and 20) and execute jumps to other parts of the program that switch an output state if this is the case. The code for a Low state starts at line 200, while that for a High state starts at line 250. Besides this, the counter is reset to zero at these locations.

The important feature of this program is the testing of conditions that either cause certain jumps to be executed or avoid making precisely these jumps. IF… THEN… ELSE… thus means that if the specified condition is satisfied, then jump to line such and such, and if it is not, then simply execute the next line. The condition is given in line 120 as `INP=1`. This means that here a comparison is made, with the result of the comparison being true if the input had a High state at exactly the same time as it was sampled in the previous line.

The program contains several comments, introduced by ‘`REM`’, that are intended to increase the readability of the program. A line may contain more than one instruction, but the instructions must then be separated by a colon (:). Basic makes it easier to search for errors, since it is possible to ‘momentarily’ insert a `PRINT` instruction at any desired location in order to view whatever intermediate results may be present. In line 150, you can see the remnants of such a test. In order to see whether the actual counter was working, `PRINT COUNT` was added. This made it very easy to check whether input pulses were being correctly evaluated. Our first version of the sample program had an error that could no longer remain hidden once this instruction was added. The error arose because during the program development, we forgot to reset the counter to zero (line 270), with the result that the comparisons with specific counts (lines 160 and 170) led nowhere. Once everything was in order, we could ‘comment out’ the `PRINT` instruction using `REM`, which means that it is practically disabled. Of course, we could also delete it, but if exactly the same output instruction again proves to be useful in some future testing, we can simply delete ‘`REM`’ and thereby restore the old state.

The interesting question now is, what is the upper frequency limit that this counter can achieve? To answer this question, we connect a function generator to Port P1.0 and observe the P1.1 output using an oscilloscope. Our first impression is disappointing. Although simple LS TTL ICs can easily handle this task at an input frequency of 50 MHz, with this Basic counter the limit is already reached at 50 Hz. The program is thus around a million times slower. If the input frequency is raised above 50 Hz, input pulses are missed since at the time that the pulse should be sampled, the program has not yet returned to the line that samples the input. Still, the program can count pulses somewhat faster than a person can.

Comparing the program with a purely electronic solution or a manual solution is actually not particularly fair. The proper question is instead, in which cases can the Basic solution be the proper solution? There are potential applications wherever relatively slow results must be counted and evaluated. For example, we could build a coil-winding machine for which we want the motor to be switched off after exactly 1650 turns. In this case, the Basic program proves to have the advantages that it can be quickly and easily modified and that the counting range is not limited by hardware.

Autostart for BASIC-52

Assembler programs have the advantage that they are autostart-capable with the Flash Board. This means that they can automatically start to work on their own each time the power is switched on. Unfortunately, this is not directly possible with BASIC-52, since the actual program is stored as data bytes in the RAM. When the power is switched off, everything in the RAM is lost. Even if we use a battery-backed RAM, the previous program would not automatically start. Instead, it would be fully deleted by the Basic interpreter, since BASIC-52 likes to start with a completely empty RAM.

However, from the very beginning the developers of the interpreter provided a function that allows a program to be held in EPROM in the memory region starting at 8000h. This capability can also be used with a supplementary EEPROM located in this memory region. There are two possible ways to implement this extension. The first is to use the system bus available at connector K8, while the second is to solder a second socket ‘piggyback’ on top of the RAM. Almost all of the leads can be connected one to one, with only the $\overline{CS}$ (pin 20) and $\overline{WR}$ (pin 27) leads requiring special treatment (see **Figure 4**).

The RAM occupies the address range 0–8FFFh. The address decoding is very simple, since it only amounts to connecting $\overline{A15}$ to the $\overline{CS}$ pin of the RAM. Whenever $\overline{A15}$ is High, which means for memory accesses above 8000h, the RAM is inactive. In accordance with this simple arrangement, address signal A15 must be inverted before being applied to the $\overline{CS}$ input of the supplementary EEPROM. In this way, the EEPROM is blocked in the lower memory range and occupies the range above 8000h.

From experience with EEPROMS,

it appears that there is a good chance of their contents being altered when power is switched on. This is presumably due to a brief pulse on the $\overline{\text{WR}}$ input. If this happens, the EEPROM is of little use, since the autostart will only work if stored program is free of errors. The proper solution is a jumper that can be used to block accidental programming of the EEPROM. **Figure 4** shows the full modifications for a 28C64 8-kB EEPROM, including an inverter and write protection.

The original version of BASIC-52 can program its own EPROMs, but this requires special hardware that is not present here. However, that does not present a problem, since the programming process for the EEPROM is so simple that it can be carried out using a small Basic program. The utility program listed below can do the job. It is added on top of an existing program (in this case, the program in lines 10–40) and started using `GOTO 9000`. It simply copies all program bytes from the RAM to the EEPROM, starting at address 200h in the RAM and address 8011h in the EEPROM. Three additional bytes provide a problem-free autostart by supplying important information, such as the baud rate setting, to the operating system. For correct operation, it is also important to use `MTOP` to limit the upper boundary of the RAM.

```
10 FOR N=0 TO 255
20 PORT1=N
30 NEXT N
40 GOTO 10

9000  N=200h : E=8011H
9010  D=XBY(N)
9020  PRINT N,D
9025  XBY(E)=D
9030  N=N+1 : E=E+1
9040  FOR T=1 TO 10 :
        NEXT T
9050  IF D<>1 THEN
        GOTO 9010
9100  XBY(8000H)=32H
9105  FOR T=1 TO 10 :
        NEXT T
9110  XBY(8001H)=0FFH
9115  FOR T=1 TO 10 :
        NEXT T
9120  XBY(8002H)=0DCH
9125  FOR T=1 TO 10 :
        NEXT T
9130  XBY(8010H)=055H
mtop = 8191
goto 9000
```

As soon as the EEPROM has been programmed, you can use the `ROM` command to switch to the upper address region. The stored program can now be started in a perfectly normal manner using `RUN` and stopped using Ctrl-C, but it cannot be edited. You can use `RAM` to switch back to the normal memory region. In this manner, it is effectively possible to have two separate programs available in the system.

Each time the supply voltage is reapplied, the program currently stored in the EEPROM is automatically started. If this program is to be used permanently, the programming jumper should be removed in order to prevent accidental modifications to the program.

This concludes our brief introduction to working with BASIC-52. In the next instalment, we will work with the C language. This will complete our comparison of the three most important programming languages.

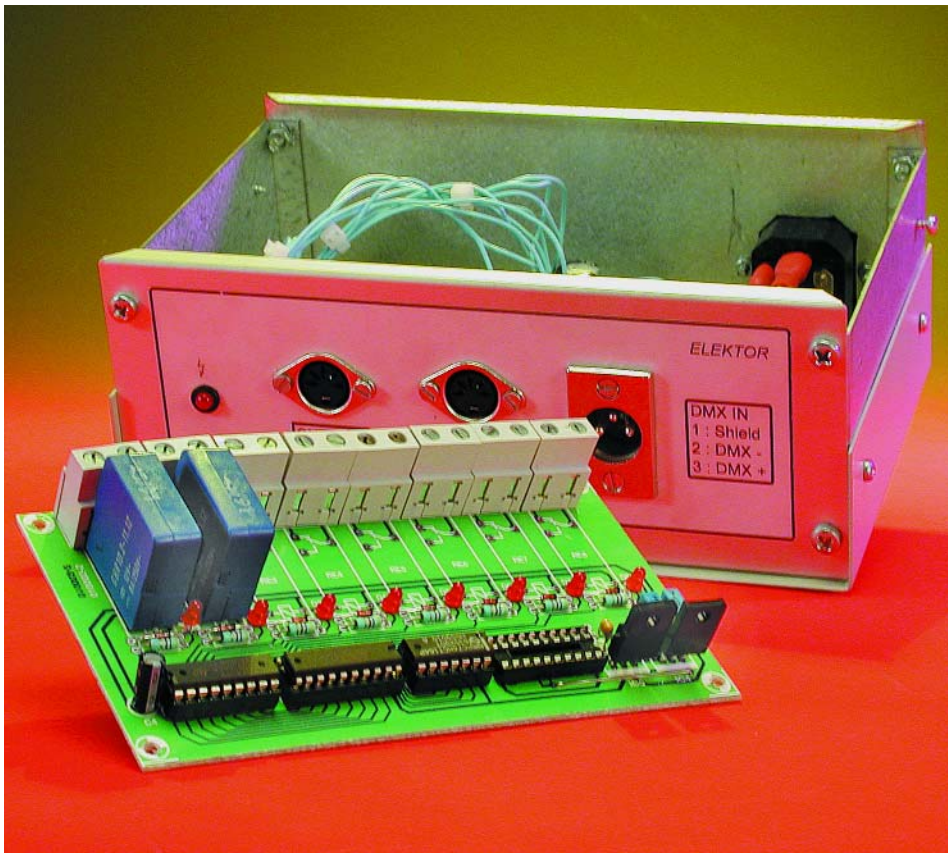
(010208-4)

Relay Extension for DMX Demultiplexer

eight DMX channels for eight relays

Design by B. Bouchez

The DMX demultiplexer, which was described in *Elektor Electronics* recently, enables you to drive eight analogue outputs with a control voltage from 0 to 10 V, a standard that is used for many lighting installations. For less elaborate equipment, such a control voltage is not actually necessary, since the control is limited to either 'on' or 'off'. This is the intended application for this extension circuit which provides eight relay outputs.



The remote control of lighting installations with voltage or current is still very common nowadays, despite the existence of DMX512. In all likelihood either because the design is quite old (dimmers mostly), or because it is equipment where the additional costs of a digital interface would be unjustifiable (stroboscopes, smoke generators). This was one of the motivations for designing the 8-channel DMX/10-V interface, which is able to control such equipment directly (refer to *Elektor Electronics* November 2001).

There is also equipment out there that requires even less elaborate control, in the form of a simple contact closure. Controlling such equipment with the demultiplexer is not very complicated (a transistor circuit that drives a relay when the voltage is greater than some threshold will suffice). Such a solution is not very attractive from an economic perspective however, particularly because of the presence of an 8-channel DAC, which has effectively become unemployed.

The author, therefore, has designed this small extension that makes the DAC and accompanying circuitry unnecessary and replaces it with an inexpensive digital circuit that drives a relay instead.

Before we move on, we have to point out two important facts. Firstly, the use of this interface precludes the option of using the DAC (it is either one or the other). It is not possible to mix the analogue outputs from the DAC with the digital outputs from the relay extension in the same multiplexer.

Secondly, readers who have already built the analogue version need not make a single change to the circuit. All necessary modifications have already been incorporated in the published version, both from a programming perspective as well as from a hardware perspective. It is actually enough to remove the DAC8800 from its socket, replace it with the ribbon cable connector from the relay extension and to flip a configuration DIP switch to indicate to the program that we wish to use the relay extension.

The circuit

As can be observed from **Figure 1**, this relay extension can definitely not be accused of being terribly complicated! There is hardly anything there, apart from some very common ICs.

The interconnection to the main circuit board is simply accomplished with a ribbon cable, fitted with a DIL-header that plugs in the location previously occupied by the DAC8800 on the demultiplexer PCB. The ribbon cable attached to this DIL connector carries all signals, including the power supply, which are necessary for the extension circuit.

Normally, the DAC8800 is driven with a serial data stream from the microprocessor. IC1 is present to convert this data stream into a useful format: 8 bits parallel.

It is the task of IC2 to latch the data on the data lines at the instant that the microprocessor communicates with IC1. Once the 8 bits are loaded into IC1, signal LD 'freezes' the parallel data obtained.

The outputs from IC2 are not capable of driving a relay directly. This is why an octal buffer IC type ULN2803 sits between the relay coils and IC2.

Each relay output is also provided with an

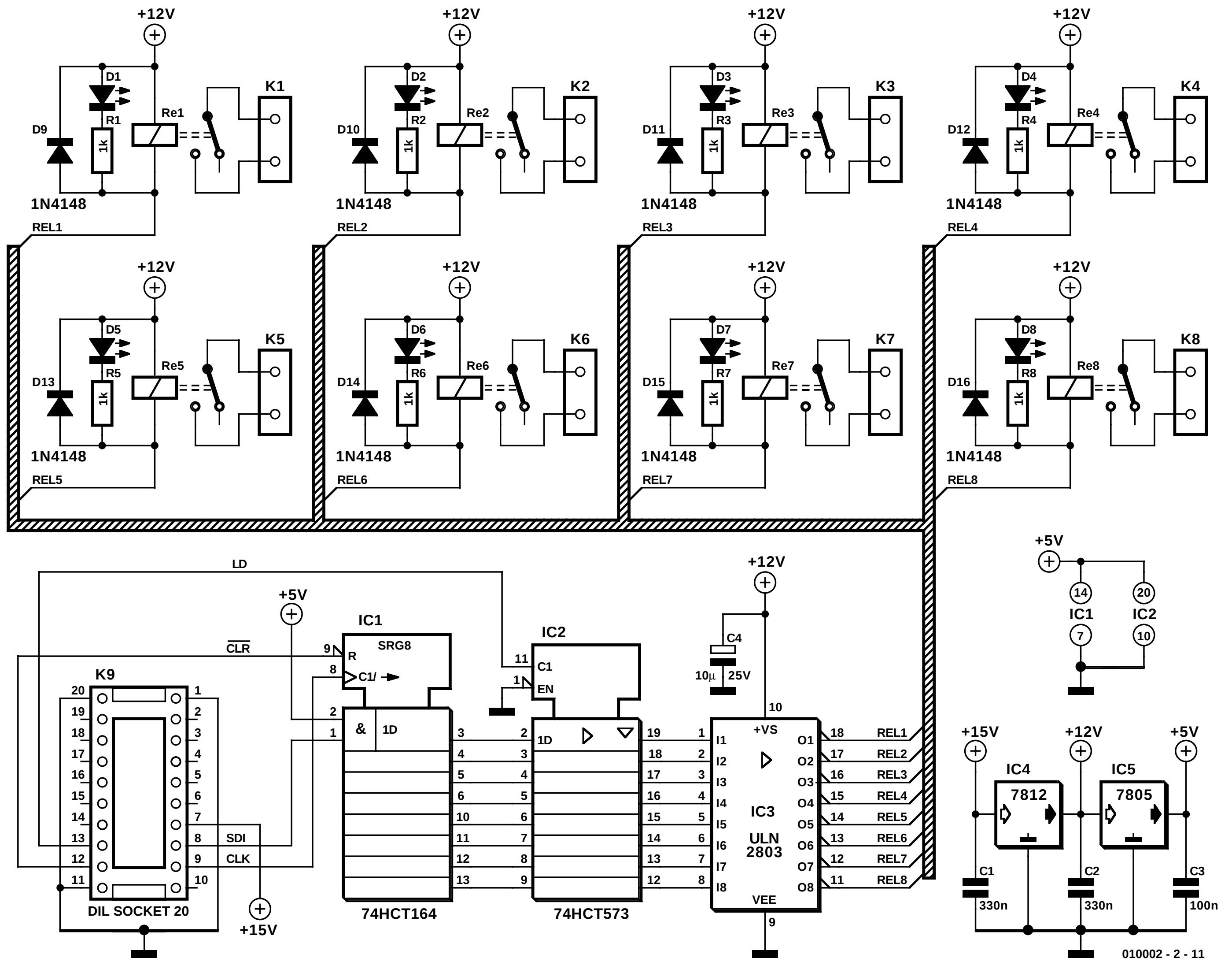


Figure 1. The schematic for the relay extension to the DMX-demultiplexer shows that it is a very straightforward design.

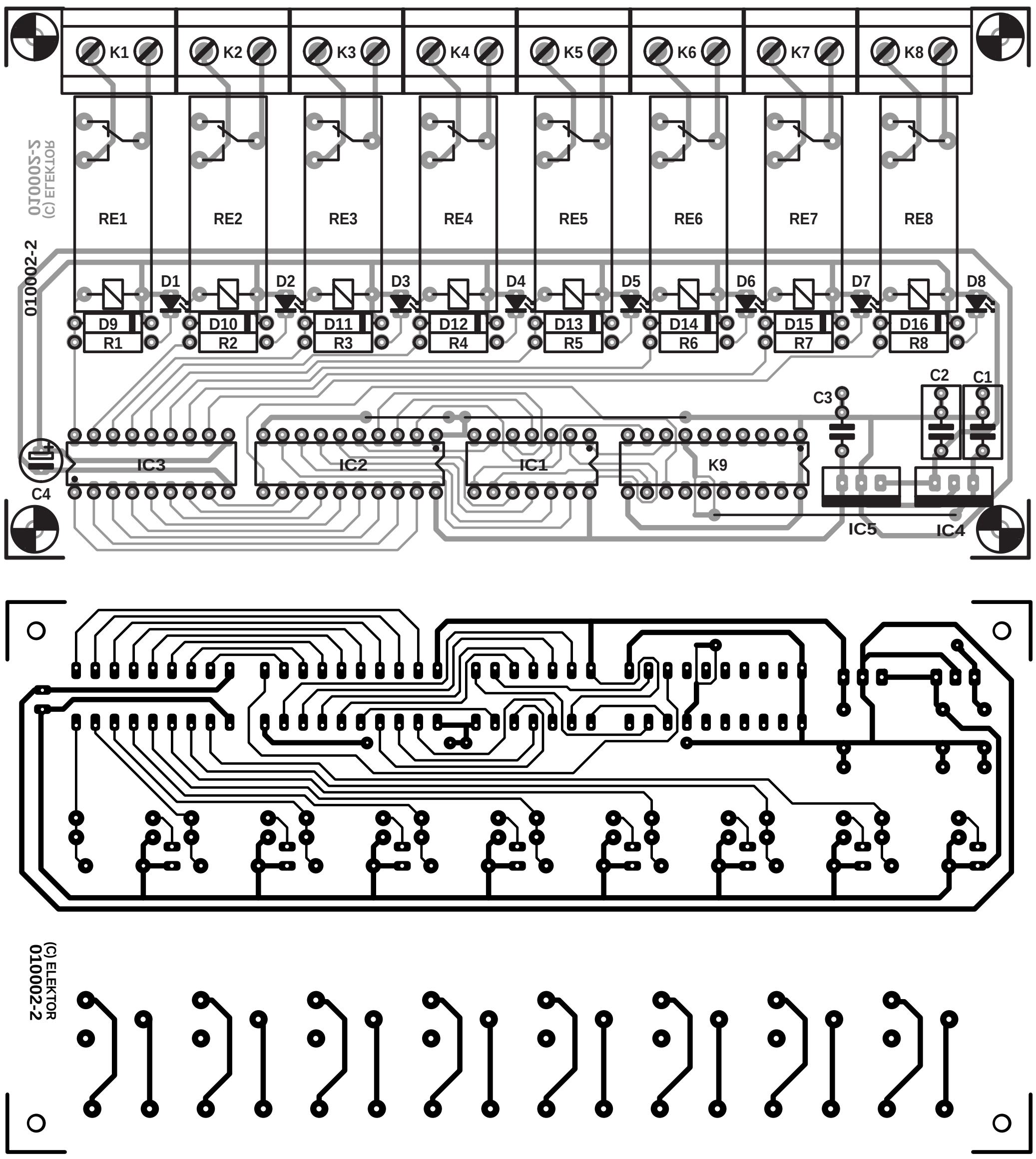


Figure 2. The circuit board for the relay extension contains a number of wire links that must not be forgotten.

LED to indicate the status of the corresponding relay.

The power supply is of a very simple design. Its only task is to convert the 15 V originally intended for the DAC into a voltage that's appropriate for the relay coils (12 V) one of 5 V for the logic circuitry.

Construction

The assembly of the circuit is just as straightforward as the schematic.

As is usual, the ICs are fitted in good quality sockets, after you have checked that the correct power supply voltages are present on the proper pins.

The cable interconnecting the main PCB with the relay extension is best made with DIL20 insulation-displacement connectors (IDC) and a short length of ribbon cable.

COMPONENTS LIST

Resistors:
R1-R8 = 1kΩ

Capacitors:
C1,C2 = 33nF
C3 = 100nF
C4 = 10µF 25V

Semiconductors:
D1-D8 = high-efficiency LED
D9-D16 = 1N4148*
IC1 = 74HCT164
IC2 = 74HCT573
IC3 = ULN2803
IC4 = 7812
IC5 = 7805

Miscellaneous:
K1-K8 = 2-way PCB terminal block, lead pitch 7.5 mm
K9 = 20-pin IC-socket
Re1-Re8 = E-Card relay, 12V, type V23057 B0002 A101 (330 Ω) (Siemens)
Length of 20-way flatcable with DIL20 IDC connectors
PCB no. 010002-2

* D9-D16 may be omitted because the ULN2803 has internal flyback diodes.

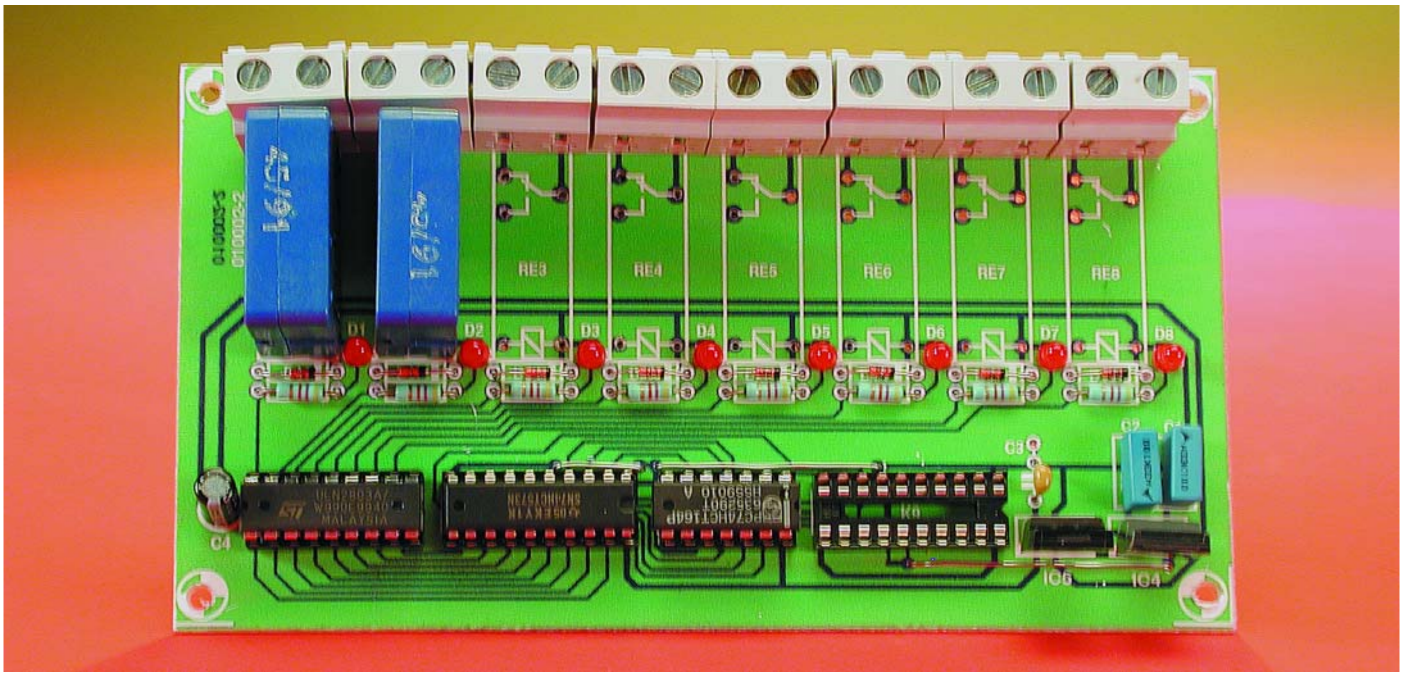


Figure 3. The prototype version shown was populated with just two relays. You are of course free to adjust that number to suit your requirements.

If you are unable to obtain this type of connector, it is also possible to use DIL plugs (intended for soldering components such as resistors and capacitors to) instead. The connections between these two plugs that will function as the interconnecting cable will then have to be built one wire at a time.

The only other thing that requires special attention are the relays Re1 through Re8. The footprint for the relay on the PCB is relatively standard and provides a certain amount of freedom as to the actual model relay that you can use. It is not mandatory to use the exact relay listed in the parts list. However, it is a requirement that you use relays with a low coil current or you will run the real risk of unexpected behaviour when all relays are driven simultaneously and the supply voltage sags as a consequence. We would like to make a concluding remark by saying that it is not necessary to fit all relays — you only need to fit the number required for the application you have in mind.

Connecting up…

Once the wiring for the relay extension has been completed and you have carefully checked it, it can be connected to the socket for the DAC8800, IC4 of the demultiplexer with the DIY ribbon cable. The opamps on the output may also be omitted. One end of this cable is plugged into the empty socket on the extension PCB (refer to illustrating photograph) and the other end is inserted in the socket originally intended for the DAC.

Before applying power to the demultiplexer we have to form the program in the microprocessor that something has changed and that we are not driving the DAC8800 but are using the relay extension instead. To achieve this it is enough to set DIP switch S1-7 on the main PCB in the ON position.

Next, you are ready to apply power. None of the relays should energise. If this happens nevertheless, then carefully check your work again before continuing.

To control the relays you require a DMX Master Device capable of transmitting commands to the channels occupied by the multiplexer (refer to the section about the selection of channels with S1-1 through S1-6 in the original DMX Demultiplexer article).

To activate a relay it is necessary to transmit an instruction for the corresponding channel with a value greater than or equal to 128. To deactivate the relay, issue an instruction with a value between 0 and 127.

…and operating

We leave the practical use of the ‘relay’ version of our DMX Demultiplexer to the reader/builder. In most cases, it will be sufficient to connect the two wires from the equipment that needs to be controlled, which are usually connected to a switch, to the relay contacts instead.

It is usually quite straightforward to modify existing light fittings that are not remotely controllable with an additional relay to provide them with this feature. Note that the relays in our extension are not capable of switching too large a load.

If you need to switch large loads, we recommend that you use an appropriate external relay, which in turn is controlled by the relay on the extension PCB.

In all cases where large powers are being switched (either via an external relay or directly by Re1 through Re8) you will need to add a so-called ‘snubber’ network to dissipate the energy that is generated when the contacts are opened.

It is easy to make such a network yourself by connecting a 0.5-watt resistor, with a value of several tens of ohms, in series with a capacitor of about 220 to 330 nF/630 V. This network is then connected directly across the switching contacts of the relay.

(010002-2)

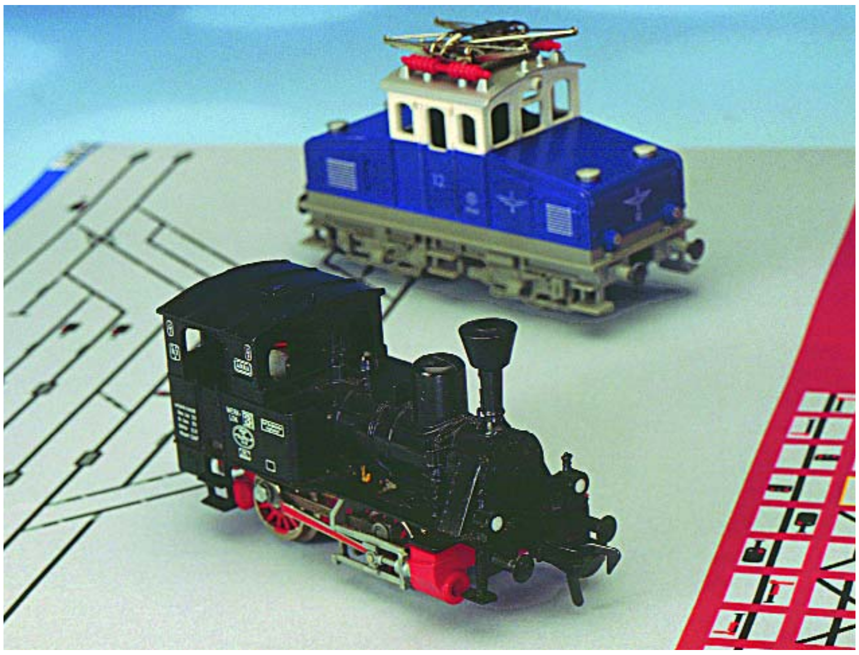
EEDTS Pro 1.2

new software, new features

By Steffen van de Vries

It's been a while since we had anything new to report on EEDTS Pro, our popular model train control system. However, considering the large number of extensions that the author has implemented in this new version, the long development time can certainly be considered to be justified. Thanks to intensive consultation with user groups, there was a good idea of the direction in which EEDTS Pro must further develop.

The wish list that was generated from consultation with the user groups, which can also be found in the EEDTS Pro book, has now been fully transformed into new hardware and software that will be described in two articles.



EEDTS Pro software: a brief summary

First let's look at the EEDTS Pro software. The most obvious extension can be found in the program lines for train control.

Functions and speeds can now also be entered into program lines, even for trains without EEDTS Pro infrared train detection. An important capability in this regard is virtual address tracking, by means of which the decoder address and location of every locomotive is always known, independent of the brand of decoder used.

This virtual address tracking is implemented without any need to use separate instruction tables for individual trains (chained sequences), and it can be implemented using a minimal number of lines in the programming mode. It is even possible to continue using a conventional electrical block system on the free track.

EEDTS Pro controller

EEDTS Pro was primarily developed as a computer interface. However, since the control unit can be constructed very quickly and easily and directly provides eight manual controls using the extended Motorola format, the controller is also popular with stand-alone users (particularly those with large-gauge systems who use it with the heavy-duty EEDTS booster).

The controller does not have any difficulties driving the locomotives. However, problems arise when there are more than eight locomotives on the track or when turnouts ('switches' in the US) must be changed over. The new controller solves these problems, since it can be started up in a special mode that allows the locomotive addresses to be modified and it allows keypads to be connected for operating turnouts and signals.

Furthermore, in collaboration with Jürgen Freiwald of Railroad & Co, the instruction set has been extended to make the controller

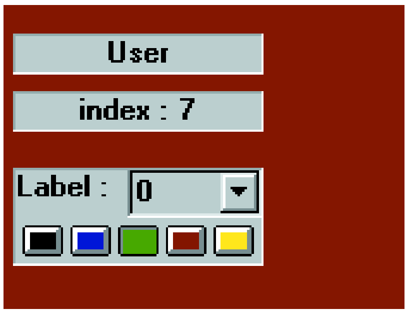


Figure 1. Every symbol in the Label palette can be assigned a number using up to five colours.

more accessible to people who want to write their own control software (for professional or hobby purposes).

A very important consideration in the further development of the controller was readout speed for the return signallers. In the old version of the controller, this was already implemented in a very fast manner, but there was a limitation in how fast the PC could read these units. In the new controller, this speed has been increased by a factor of 16, and the controller has an instruction that can be used to query which unit has experienced a change of input state since the last readout.

New hardware

The new controller does not require a new printed circuit board; the existing controller board is perfectly satisfactory. The activation of the new functions (such as addressing and keypads) was already taken into account in the design of the circuit board. Even reporting back a track short circuit (which is also a new controller function) does not require any changes to the board.

However, a new circuit board has been developed for setting the addresses of the manual controls. Using this circuit board, an eight-position switch can be used to modify the address of each manual control. It is also possible to build a *de luxe* version incorporating eight of these circuit boards, which allows the address of each of the eight manual controls to be set directly and displays the address of each controller using a seven-segment display.

The software

General issues

Problems with processor speed
In the computer world, the pace of progress is fast. When EEDTS Pro was first developed, processor speed had just passed the 100-MHz mark. Nowadays we take speeds of 1 GHz or more for granted. Of course, the task of model railway control will not generally be entrusted to the fastest machine available, but it will eventually happen – and it turns out that the ‘old’ EEDTS application stops working at around 400 MHz.

The solution to this problem is just as simple as its cause. With relatively slow processors, the number of times that the serial port is polled is always sufficient to allow the controller to respond, but with a fast processor the maximum number of queries is reached before a response has been received from the controller. This results in a time-out. The solution is simply to increase the number of times that the port is queried.

Program speed
With the new software release, real-time control of trains plays a significant role, which means that the slow generation of the (graphic) track layout creates a problem. During the generation of the track layout, so much of the processor’s capacity is used that it is not possible to respond to return signallers while the tracks are being generated and released for use. In the worst case, this can result in delays of several seconds.

In order to eliminate this problem, a complete change has been made from graphic calculation to bitmap modification. For example, whereas previously the complete 27 × 27 pixel symbol was modified in order to place a dashed yellow line, now only the two yellow areas are modified. This not only results in an impressive increase in speed, it also makes it possible to attach numeric labels to user-generated symbols for turnouts (switches) and signals.

One of the consequences of this change is that we must try to have as much commonality as possible. Consequently, the return-signalling and detection buttons now have the same shape and differ only in colour (grey or white).

Compatibility
The new software is downwards compatible, although it is a good idea to restart the software after loading a layout generated using an older version.

The new software can also be used to drive the old controller. However, in this case it is recommended not to use any real-time control commands in the program lines.

Program modes

The changes have been grouped here in terms of menu selections to provide the most comprehensible arrangement.

Build control table
The ‘Build’ window is the only one that has not been changed.

Decoder addresses
The most important change can be found in the lower left corner of the window (**Figure 1**). Previously, both the index and the ‘nx index’ were displayed next to each symbol (which led to a certain amount of confusion); now only the index is displayed as a reference for the symbol.

Also, in the program lines the index number is now shown where the nx reference was previously shown.

A label palette has been added. On this palette, each symbol (including a straight section of track, for instance) can be assigned a number using five different colours. The range is limited to two digits (1–99), due to considerations of available space and legibil-

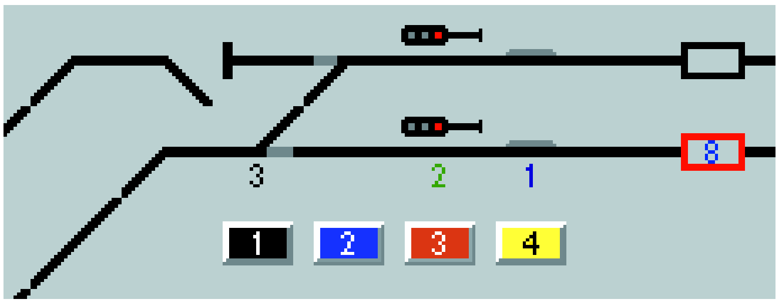
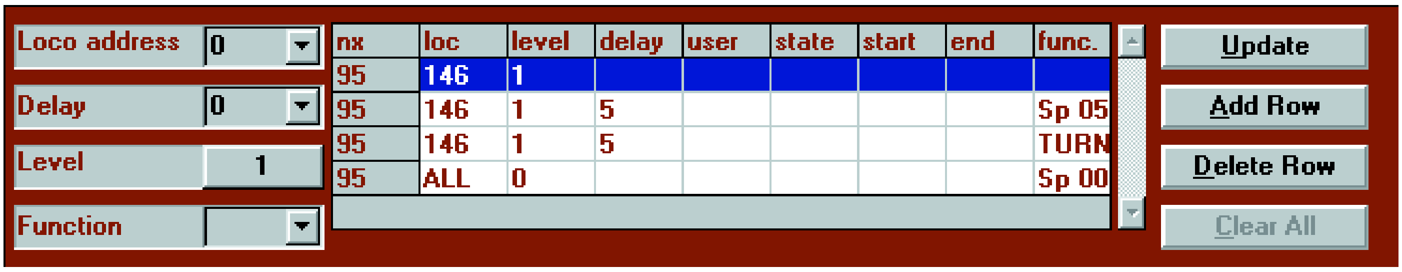


Figure 2. Now it is also possible to use English turnouts with double-coil actuators.



nx	loc	level	delay	user	state	start	end	func.
95	146	1						
95	146	1	5					Sp 05
95	146	1	5					TURN
95	ALL	0						Sp 00

Figure 3. Level 0 can now also be used in the programming window.

ity (see **Figure 2**).

This palette can also be used to assign colours to the switches and pushbuttons, which can also be assigned reference numbers.

It is now possible to also use English turnouts (switches) with two-coil actuators. The addressing has been expanded for this purpose, and up to four address/data fields can be filled in. An English turnout with a two-coil actuator needs only two address/data fields, so the software can determine which type it is dealing by examining the values assigned to the fields.

The number of possible decoder addresses has been extended to 240, so all five address selection lines of the MC 145027 can be used. This extension also covers the previous four extra functions, so pushbuttons can be used here as well.

In addition, all of these addresses can be set using the EEDTS switch decoder.

It is also possible to use other types of decoders, although this requires certain modifications.

Secured track sections

The strict requirements for the definition of secured track sections have been dropped.

This means that it is now possible to define a secured track section that is not fully closed. This is particularly important when such sections must cross each other.

A disadvantage of the elimination of software arbitration is there is no checking of conditions such as whether a turnout (switch) is in the correct position, so you have to keep your eyes open!

The program

The part that has been changed the most is the screen display.

In the original version, the screen display was only intended to be used to define program lines, but in the new version virtual address data can also be entered in this window. This is done by using the left mouse to click on a red button (track occupied) for a return signaller or detector button, following which the address of the train responsible for the activated button can be entered.

The right-hand mouse button is used to activate the program lines linked to the selected button.

In the programming table, the biggest visible change is a column in which locomotive functions can be specified. The most important control parameters to be found here are speed, reversing and five functions.

One practical option is restoring the speed value set by the soft controller (after a stop, for example).

These options can be entered in a programming line together with a locomotive address. The priority for processing alternatives is still determined by level numbers.

If a particular function must be executed immediately on activation of a return signalling or detection button, the program line is not allowed to contain a delay or a secured track section. If a program line contains a secured track section, the function will be activated only after the secured track section has been released.

Previously, '1' was the highest priority level, but level '0' has been added in the new version. At this level, only locomotive addresses can be specified along with a locomotive function. A particularly powerful function at level 0 is provided by the combination of locomotive address 'ALL' and speed level '0' (**Figure 3**). This program line can be used to stop all trains for an unsafe signal without using 'dead' sections.

Since '79' is the program address for the super locomotive decoder, this address is not available for selection.

As already noted above under 'General issues', this program design provides a maximum of possibilities with a minimum of program lines.

Soft controllers

A number of extensions have also been implemented for this menu item (**Figure 4**). First, the address range has been extended to 255, so address coding is consistent with Uhlenbrock (and perhaps we can thus arrive at a bit of a standard).

The old version had only two types (normal and extended), but a third type (mixed) has been added in the new version. The 'normal' (old) option is used for driving old-format locomotive decoders.

Since it was also possible to control functions using the old format, it is now possible to specify these functions. Since the old-format locomotive decoders do not have function outputs, the Conrad function decoder or the vehicle decoder can be used for this purpose. The control corresponds to the function buttons on the older model- 80 controller.

The 'extended' option (new) is intended to be used to drive Motorala-2 decoders (which, oddly enough, have no relationship with Motorola), both for speed and for functions.

The mixed-format option has come about because it is evident that trains are being built using a Motorola-1 decoder plus a Motorola-2 decoder for the extra functions (the Märklin ICE 3). This option can only be used in situations in which, for example, an old-format Delta decoder is combined with a Conrad FD3 Motorola-2 function decoder.

Operation

It only makes sense to be able to address up to 255 trains if it is also possible to run 255 (!) trains, so the number of controllers must also be extended from 80 to at least 255.

We couldn't quite make it to 255, but 240 is still a quite respectable number. In order to provide the necessary space for the 12 buttons needed for this purpose while still satisfying the basic requirement that everything fits a screen resolution of 640 × 480, it was necessary to rearrange things. Nevertheless, the end result is still easy to understand.

No other visible changes have been made with regard to operation, although train numbers can now be seen on the detection buttons thanks to the use of virtual addressing, and these numbers are passed on when the trains are put into motion.

Address tracking is governed by secured track sections. If a train is directed from A to B by opening secured track sections from A to B (either manually or via the program lines), the program knows that when button B becomes activated, the train is coming from A and it can pass the corresponding address from A to B.

The situation is different if no secured track sections are opened and the train moves from A to B. In this case, an error-free transfer can only be assured if it is certain that when B becomes activated, it can only mean that the train is coming from A. For EEDTS Pro, this is true if there is only one defined secured track section going to B and it comes from A. In practice, this can be the case with a section of track that is always travelled in the same direction without branching, such as an automatic block.

Manual controls

Due to the extension of the address range, manual controls can now also be set to addresses within the full range of 1 to 255.

Programming the
super locomotive decoder

Programming the DIL version of the super locomotive decoder has not been modified, although it is now possible to program an address within the extended address range.

However, a design feature of the decoder model that has been supplied up to now excludes a number of addresses (although it is possible to program all addresses in the latest model). The addresses that can be used are:

83
123
145–190
193–255

This means that more than 180 addressed can be used.

Programming the model 'N'
super locomotive decoder

This decoder model, which can only be obtained from the EEDTS Pro website, can indeed be programmed over the full address range, and with this model it is even possible to program the individual speed levels.

Programming Uhlenbrock decoders

The programming of the Uhlenbrock (Motorola) decoder features three phases:

1. Switching the decoder from the operating mode to the programming mode.
2. Programming the decoder characteristics.
3. Termination and returning to the operating mode.

In the Uhlenbrock programming window, select a decoder address (this is set to 1 when it leaves the factory) and then press the Start button. After the front lights flash a few times, the decoder enters the programming mode and the buttons are enabled for setting the address, maximum speed, minimum speed, rate of acceleration and braking deceleration.

In order to modify the address, you can enter a new address and then actuate the programming button next to the address entry field.

To set the acceleration or deceleration, move the slider to the proper position and actuate the adjacent programming button.

A slightly different procedure is used to set the maximum and minimum speeds. In this case, you first have to actuate the adjacent programming button. After a few seconds, the button will turn green and the locomotive to be programmed can be controlled using the slider. Run the locomotive at the proper maximum or minimum speed, and while it is running actuate the programming button again to store the setting in the decoder.

Once all the programmable functions have been stored in this manner, you can exit the programming mode by means of the End button.

Programming Lenz decoders

Lenz decoders actually have the same three cycles as described above, but a pushbutton on the decoder is used to switch from the operating mode to the programming mode and back to the operating mode.

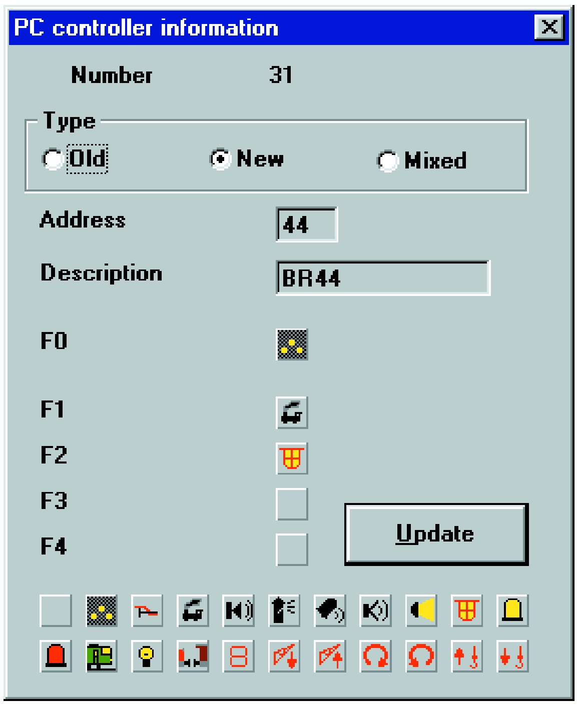
If the decoder is in the programming mode (as indicated by the front lights flashing), the Lenz programming window can be used to program the decoder. The desired values can be entered using the address entry field and the sliders for maximum speed, minimum speed, acceleration and deceleration. In each case, pressing the adjacent programming button transfers the value to the decoder.

Exit

With the old version, it was not a major disaster if you closed the program without first storing the current settings. With the new version, it is highly annoying to have to re-enter all the virtual addresses required for address tracking. Consequently, on exiting the program, EEDTS Pro explicitly asks whether you want to save the current situation.

The improvements to the controller software and the address input circuit board will be described in an upcoming article.

(010088-1)


PC controller information

Number 31

Type
Old [✓ New] Mixed

Address 44

Description BR44

F0
F1
F2
F3
F4

Figure 4. There are now three types of soft controller to choose from, and the address range has been increased to 255.

DTMF Code Lock

using clever discrete logic

Design by B. Moosmann and B. Schillo

Whether simple or ‘uncrackable’, whether constructed from discrete logic or using a microcontroller — code locks have become something of a tradition in *Elektor Electronics*. This one, however, is different!

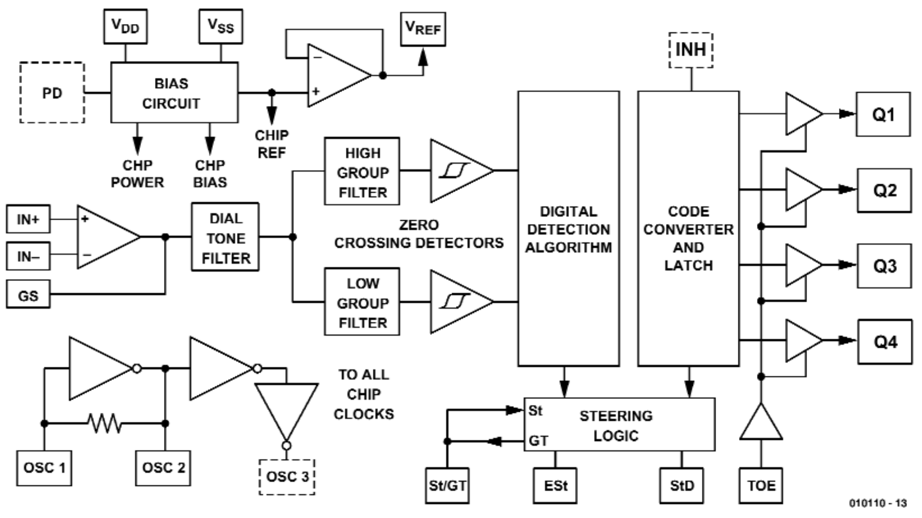


Figure 1. Internal circuit diagram of the CMD type CM8870 DTMF decoder.

This special code lock has no keyboard, but instead responds to DTMF tones, as produced by any modern mobile phone or, of course, a pocket DTMF generator. Since no-one who is anyone leaves their house these days without their mobile phone (although they might forget their front door key), almost anyone will be able to use this device.

The lock presents to the outside only a microphone capsule and four LEDs, which show the status of the unit. Their function will not be obvious to the uninitiated. The electronics consists mainly of standard components and contains no programmable devices at all.

From tones to logic levels

Our starting-point in the circuit shown in **Figure 2** is the CM8870 DTMF decoder produced by California Micro Devices (IC2), which takes the DTMF signal from the microphone after it has been processed by the op-amp at its input. The gain of the op-amp can be set from 200 to 700 using trimmer potentiometer R3. If a tone from the microphone is recognised as DTMF, the digit pressed is presented at the outputs Q1 to Q4 as a binary value. The StD signal on pin 15 goes high to indicate valid data.

Each valid entry results in the yellow LED flashing off briefly. The StD signal also serves as a clock for the type 4017 decimal counter IC8. When the first digit is entered the counter moves on one step, and the Enable1 signal goes high. The output of the decoder always represents the last DTMF digit received. If the value set on one input to the comparators IC3 to IC6 by the BCD-encoded switches matches the DTMF digit, the ‘equals’ output (pin 3) of that comparator goes high. A gate (IC7) at the output of each comparator determines whether the digits have been entered in the correct sequence, the enable signals being taken from the corresponding outputs of the 4017 counter.

If the gate is enabled, a correct digit entry will trigger the connected monostable. The period of the monostable is about 4 s, during which time code entry must be completed. When all four monostables are set, monostable IC12A is triggered via IC11A, which operates the door-opener relay for 5 s. The green LED indicates when the lock is open.

As soon as the Enable4 signal is activated, monostable IC12B is triggered, which provides a delayed reset pulse to the 4017 counter. This means that after a code (correct or not) is entered, there is a delay before the lock is ready to be operated again. After this blocked period a new code can be entered. LED3 and LED4 show whether the lock is blocked (red) or ready (yellow).

The circuit operates from a single +5 V supply, which is provided by voltage regulator IC1. When power is applied it is possible for the 4017 counter not to be in its reset state. In this case the reset button can be pressed, which resets the counter to zero.

Now for a few words on the construction and installation of the code lock. Despite the relatively large number of components, it is perfectly practical to build the circuit using prototyping board: the circuit is not critical. One exception: the wires to the microphone capsule, if it is not to be mounted on the circuit board, should be shielded and kept short. The LEDs, if mounted off the board, can be connected with ordinary signal wire. All the ICs (except the voltage regulator) should be fitted in sockets. Domestic door-opener relays usually operate from 24 V AC; 230 V versions are, however, also available. The relay used must be suitable for the voltage to be switched. Also, all the usual precautions must be observed when dealing with mains voltages.

(010110-1)

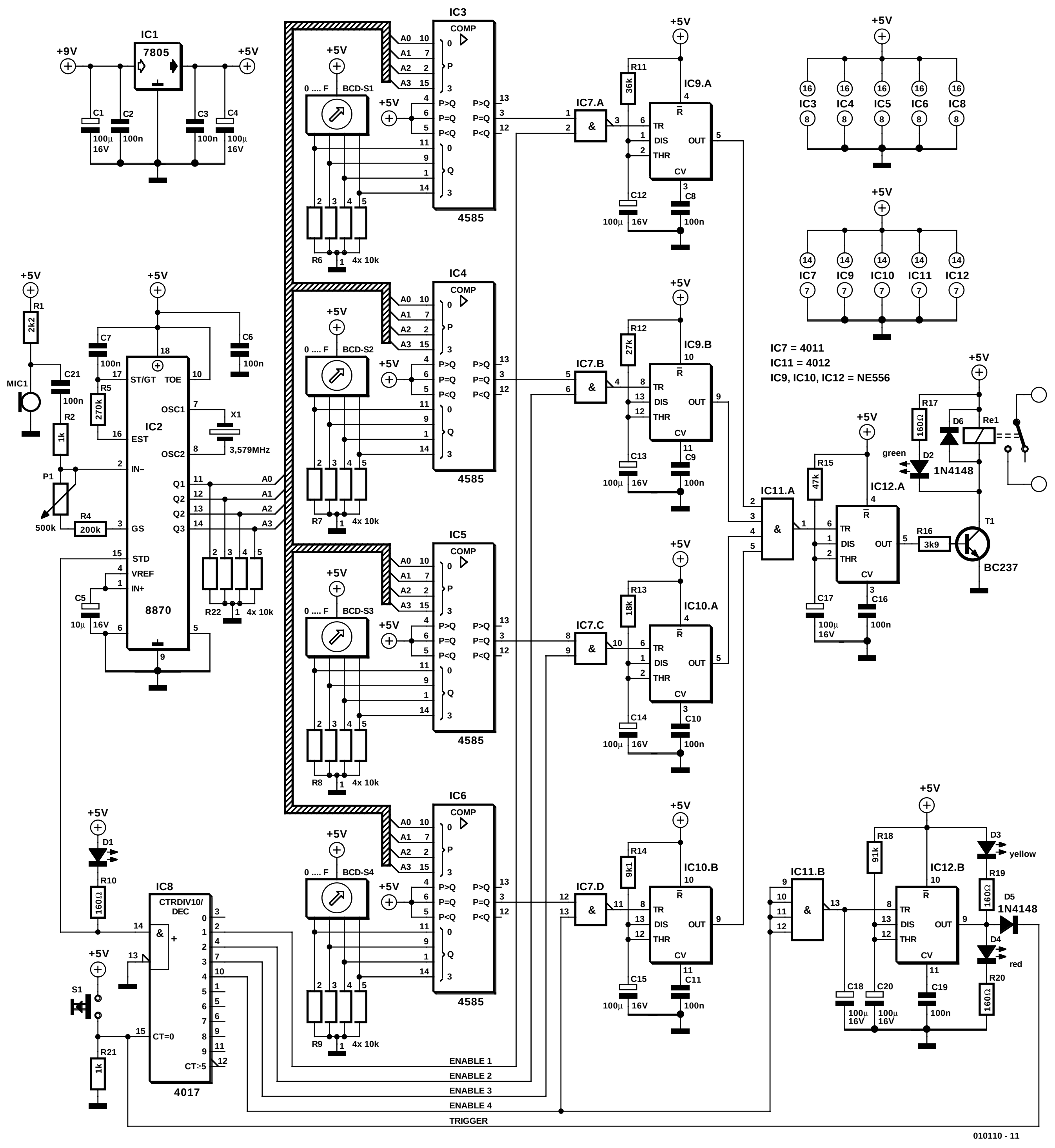


Figure 2. Circuit diagram of the DTMF code lock.

Adjustable AF Notch Filters

high Q factors in the audio range

By H. Kühne

Add two op-amps to a Wien-Robinson bridge and we can construct a notch filter whose centre frequency can be set using a single resistor. This is a significant advantage when compared with the commonly-used double-T network.

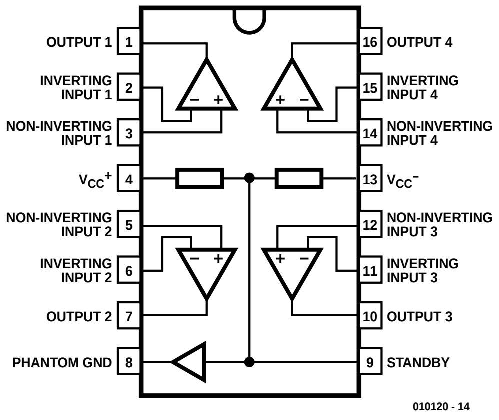


Figure 1. This quad rail-to-rail op-amp also offers a buffered virtual earth output.

The notch filter described in this article employs the ST Microelectronics type **TS925 quad operational amplifier**. This part offers rail-to-rail (ground to U_S) operation on both inputs and outputs and has the additional advantage that an internal voltage divider and buffer IC1.E generate a virtual ground at a potential U_S / 2 (**Figure 1**).

The circuit of **Figure 2** shows a modified Wien-Robinson bridge with resonant frequency f_o. IC1.B, together with C2, C4, R3, R6, R7 and R8 realise the complex branch of the bridge, while R1+R2, R5 and IC1.C form the real branch. Here, IC1.C operates as an inverting amplifier whose gain can be adjusted via R1 in the range 0.93 to 1.07 to obtain the best rejection of signals at frequency f_o. In the prototype a ratio (u_{o_eff} / u_{i_eff}) of better than –55 dB was achieved. With input signals of a frequency $f_e << f_o$, C2 and C4 have

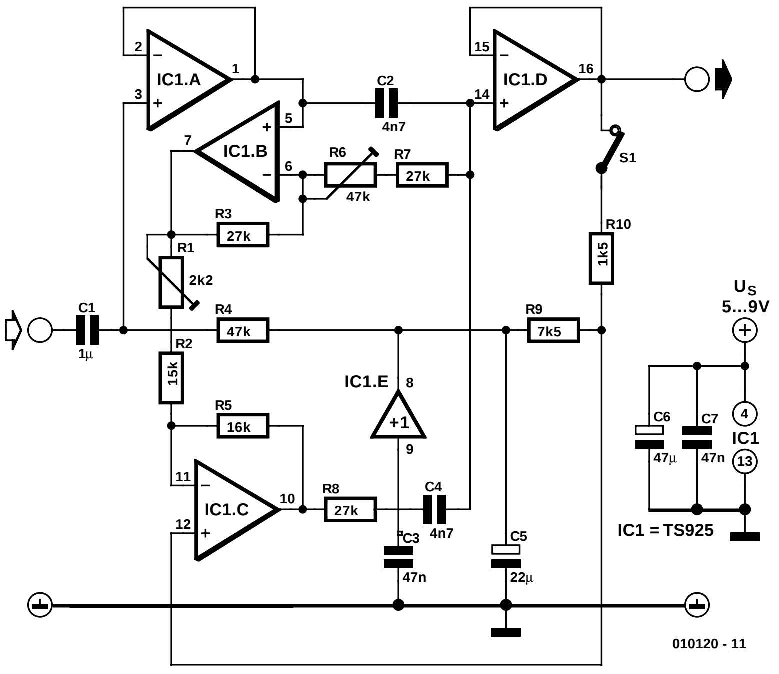


Figure 2.Simple adjustable notch filter.

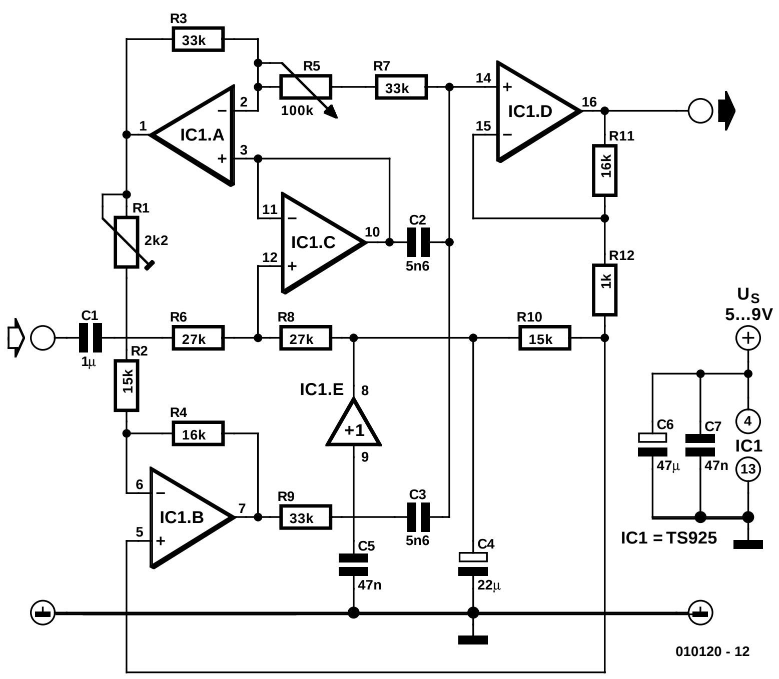


Figure 3. Notch filter suitable for higher input voltages.

practically no effect and the circuit works as a voltage follower with unity gain. In the case $f_i >> f_o$, C2 and C4 act as short-circuits, and again the desired voltage follower operation is obtained. With the requirement that R3 = R8, C2 = C4, and R1+R2 = R5, we can derive the equation for the centre frequency of the extended Wien-Robinson bridge circuit around IC1.B and IC1.C:

$$f_o = 1 / [2\pi \text{ R8 C4} \sqrt{(\text{R6} + \text{R7}) / \text{R3}}]$$

It makes essentially no difference whether (R6+R7)/R3 is greater than or less than 1. (R6+R7)/R3 ≥ 1 is, however, more appropriate from the point of view of the maximum permissible filter input voltage. Then, independent of the position of the adjustment potentiometer R6, the peak output voltages of IC1.B and IC1.C are at most $\pm\hat{u}_{IC1.B} = \pm\hat{u}_{IC1.C} \leq \pm 2\hat{u}_e$, assuming the input signal to be a sine wave. With the rail-to-rail op-amps used here and with (R6+R7) / R3 ≥ 1 we can thus allow sinewave inputs with amplitudes of up to $\pm\hat{u}_i = U_S / 4$, or in other words $u_{i\_eff} = U_S / 4\sqrt{2}$.

The ratio (R6+R7)/R3 affects the Q factor Q_S of the notch filter. With S1 open and R6+R7 = R3 we can achieve a Q factor of 0.5, which is the value which can be derived for the basic Wien-Robinson bridge arrangement. The Q factor can be increased by feeding back a fraction of the output voltage to the non-inverting input of IC1.C. For Q_S we then have:

$$Q_S = \text{R9+R10} / [2 \text{ R10} \sqrt{(\text{R6+R7}) / \text{R3}}]$$

With R9=0 Ω or with S1 open the feedback is disabled. The filter then has Q factor $Q_S = 1/2 \sqrt{(\text{R6+R7})/\text{R3}}$. On the other hand, with R10=0 Ω the Q factor becomes infinite: the arrangement becomes unstable and oscillates. The prototype values were calculated for $u_{i\_eff} \leq 0.8$ V at $U_S = 5$ V and for a centre frequency range of 0.76 kHz to 1.25 kHz.

An alternative arrangement

The alternative arrangement shown in **Figure 3** allows a pass-band gain of 1 for sinusoidal input signals with $\pm\hat{u}_i = U_S / 2$. This is achieved using an input voltage divider (R6 and R8), whose attenuation is compensated for by op-amp IC1.D with a gain of 2. With C2=C3, R3=R9 and R1+R2=R4 we have the following expressions for the centre frequency and Q factor of the notch filter:

$$f_o = 1 / [2\pi \text{ R9 C3} \sqrt{(\text{R5+R7}) / \text{R3}}]$$

$$Q_S = (\text{R10+R12}) / [2 \text{ R12} \sqrt{(\text{R5+R7}) / \text{R3}}]$$

The component values shown are suitable for sinusoidal inputs with $u_{i\_eff} \leq U_S / \sqrt{8}$, for U_S in the range 5 V to 9 V, and for a centre frequency range of 420 Hz to 834 Hz. As this is adjusted via variable resistor R5, the Q-factor of the filter also varies, as can be seen from the transfer characteristics plotted in **Figure 4**. It is interesting to note that the bandwidth of the filter $B_S = f_o / Q_S$ is independent of this adjustment:

$$B_S = \text{R12} / [\pi \text{ R9 C3 (R10+R12)}]$$

At both f_o = 420 Hz and f_o = 834 Hz a bandwidth of 110 Hz is measured; the calculated value is 108 Hz, showing good agreement between theory and practice.

(010220-1)

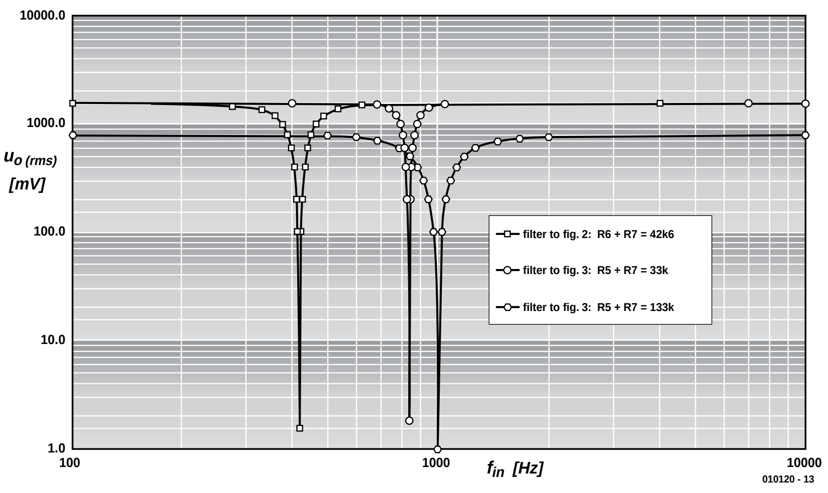


Figure 4. Frequency response of the notch filter.

8051 Compilers

freeware, shareware and demo versions

By Harry Baggen

The 8051 (and its hoard of follow-up compatible derivates) is the most widely used microcontroller in the industry. One of the advantages of working with this microcontroller is the huge amount of information on associated hardware and software. The Internet is by far the largest resource for users of 8051 derivates. In this article we will focus our attention on compilers for the 8051.

The good old 8051 and 8052 may well thank their renown to the ease of programming via the world famous BASIC-52 interpreter (which *Elektor Electronics* magazine was the first to describe and use in practical projects). These days, semiconductor manufacturers produce controllers whose core is based on the 8051 architecture, although the actual chip technology is much faster and extended with respect to the original introduced many years ago. The extended architecture may include more memory, a watchdog, an LCD interface, D-A and A-D converters and various other elements.

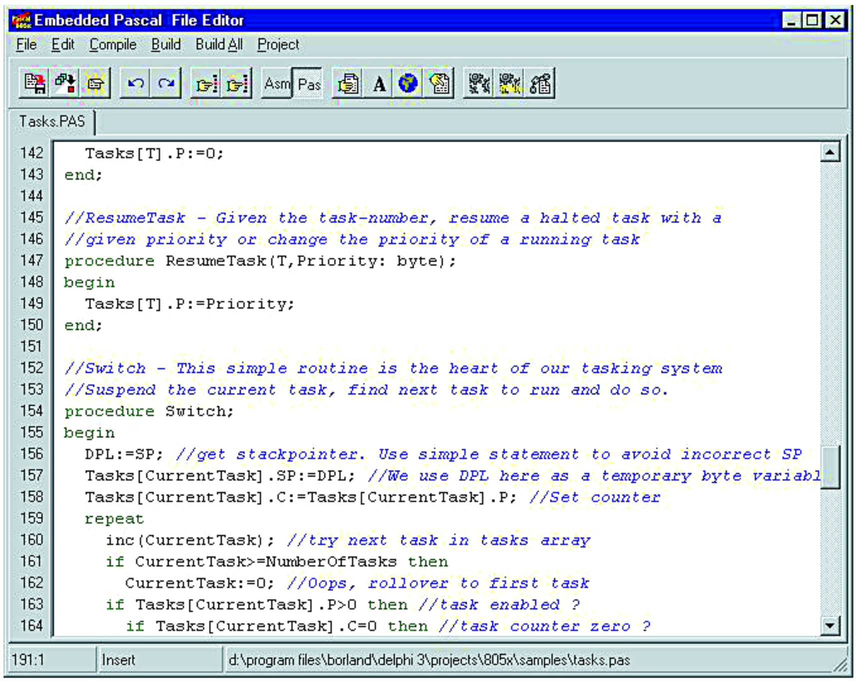
The 89S8252 found on the development board we use for our Microcontroller Basics Course is a fine example of an advanced 8051 derivate.

For many applications, BASIC-52 is simply too slow, and the obvious alternative is to write code in assembler. Alas, obvious it may be, but easy, no. A compiler represents a midway alternative, translating a number of 'legible' commands in machine code that can be copied into the microcontroller.

Although a large number of 8051 compilers is available on the market, most will set you back at least £70. A bit stiff, we'd say, if you are just curious or doing occasional programming only. Fortunately, there are also free compilers and demo versions with limitations regarding the size of the program code. Such programs may be just the ticket if you limit yourself to relatively small programs. Believe us, you will never cease to be amazed about the power of a cleverly designed machine code program of just 1 or 2 kBytes.

One of the best-known free C compilers for the 8051 is called **SDCC** [1] which is short for *Small Device C Compiler*, although some sources declare that the acronym stands for *Sandeep Dutta's C Compiler* after the name of the original developer. SDCC is a code optimising ANSI-C compiler for the 8051 and Z80 microcontrollers. Reportedly versions are being developed for other popular controllers. SDCC is available for the Windows and Linux operating systems. The program is continuously improved and extended by a number of programmers.

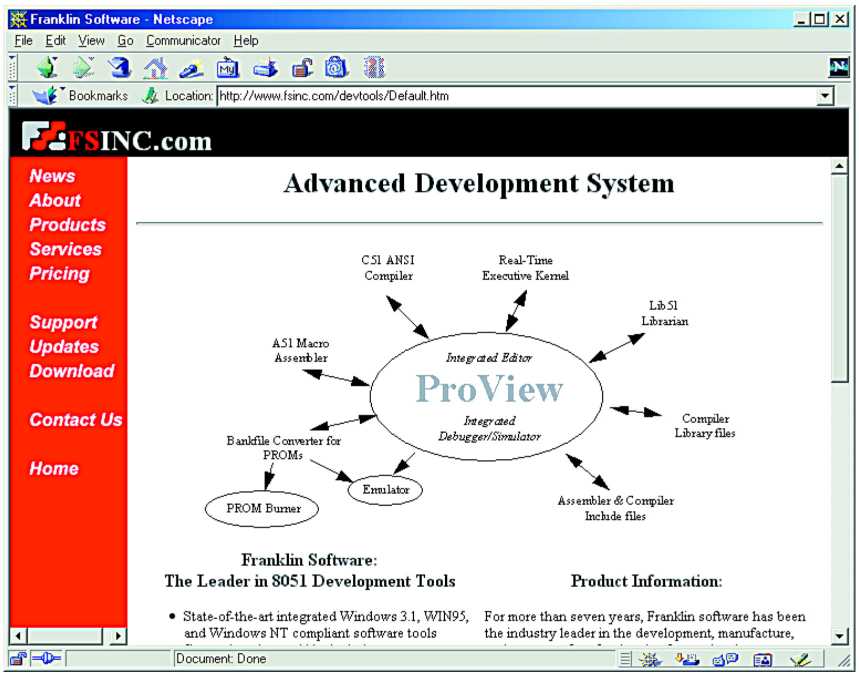
Another free compiler (which was already mentioned in the Microcontroller Basics Course) is


Embedded Pascal File Editor
File Edit Compile Build Build All Project
Asm Pas
Tasks.PAS
191:1
Insert
d:\program files\borland\delphi 3\projects\805x\samples\tasks.pas

```
  Tasks[T].P:=0;
end;

//ResumeTask - Given the task-number, resume a halted task with a
//given priority or change the priority of a running task
procedure ResumeTask(T,Priority: byte);
begin
  Tasks[T].P:=Priority;
end;

//Switch - This simple routine is the heart of our tasking system
//Suspend the current task, find next task to run and do so.
procedure Switch;
begin
  DPL:=SP; //get stackpointer. Use simple statement to avoid incorrect SP
  Tasks[CurrentTask].SP:=DPL; //We use DPL here as a temporary byte variabl
  Tasks[CurrentTask].C:=Tasks[CurrentTask].P; //Set counter
  repeat
    inc(CurrentTask); //try next task in tasks array
    if CurrentTask>=NumberOfTasks then
      CurrentTask:=0; //Oops, rollover to first task
    if Tasks[CurrentTask].P>0 then //task enabled ?
      if Tasks[CurrentTask].C=0 then //task counter zero ?
```


Franklin Software - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://www.fsinc.com/devtools/Default.htm
FSINC.com
News
About
Products
Services
Pricing
Support
Updates
Download
Contact Us
Home
Advanced Development System
C51 ANSI Compiler
Real-Time Executive Kernel
Lib51 Librarian
A51 Macro Assembler
Integrated Editor
ProView
Integrated Debugger/Simulator
Compiler Library files
Bankfile Converter for PROMs
Emulator
PROM Burner
Assembler & Compiler Include files
Franklin Software:
The Leader in 8051 Development Tools
Product Information:
State-of-the-art integrated Windows 3.1, WIN95, and Windows NT compliant software tools
For more than seven years, Franklin software has been the industry leader in the development, manufacture,
Document: Done

Reads51 from Rigel Corp. [2]. Although this compiler is actually intended for use with Rigel's own microcontroller boards, *Elektor Electronics* readers are allowed to download it for private use in combination with the Microcontroller Basics Course. Reads51 is a combination of a small-C compatible 8051 compiler, a relative assembler, linker/locator, editor, simulator and debugger/monitor. A wonderful software suite, which requires very little time to get used to.

If your are more comfortable with Pascal than with the C programming language, you may want to give **Pascal51** a try. This popular program was already mentioned in combination with a number of *Elektor Electronics* projects. Although the download sites of this program (originally developed by a Russian programmer) seem to change a lot lately, two sites can be mentioned which seem to be fairly reliable [3]. Derek Kennedy has come up with a couple of extensions for Pascal51, called Prepo, which may be found on the same page.

Embedded Pascal for the 805x [4] is a Pascal compiler featuring a 'device definition system' that allows a compiler to be adapted to almost any 8051 derivate currently available on the market. Embedded Pascal is a shareware program that enables object code with limited size to be produced.

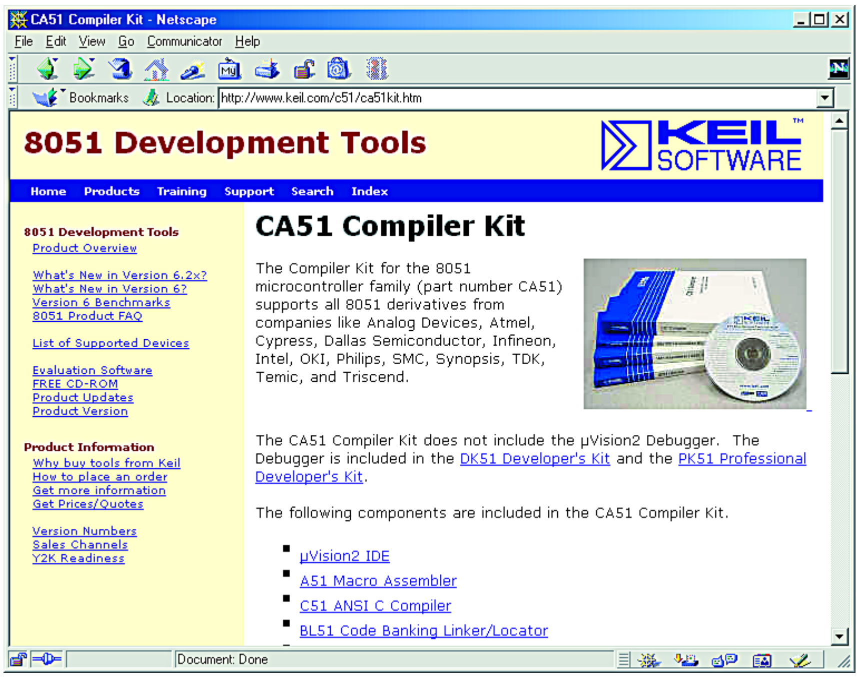
Now that we've reached the realms of shareware and demo versions, we need to return for a bit to the C compilers. A useful C compiler is contained in the evaluation version of **C51 Tools** from Keil [5]. The compiler has a code size limitation of 1 kBytes. Unfortunately, the demo version does not allow the program to start at 0000H, which means that it can not be used for single-chip controllers.

Franklin Software supplies a comprehensive programming environment for the 8051, with the **Proview** editor/debugger/simulator at the centre of things. A C51 ANSI compiler (version 6) is also offered. The suite comes as an evaluation package with a size of 12 Mbytes [5]. The limitation is the size of the object code file: 4 kBytes.

The **Raisonance** company supplies a number of programming tools for various microcontrollers, including te 8051, the Philips XA and the ST6 series. A demo version is available for the 8051 — maximum code size 4 kBytes [7].

As you can see, those of you who want to get cracking with compilers for the 8051 enjoy a wide choice of programs which are available free of charge or a as evaluation versions. Your programs, please!

(025019-1)


CA51 Compiler Kit - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://www.keil.com/c51/ca51kit.htm
KEIL SOFTWARE
Home Products Training Support Search Index
Document: Done

8051 Development Tools

8051 Development Tools
Product Overview

What's New in Version 6.2x?
What's New in Version 6?
Version 6 Benchmarks
8051 Product FAQ

List of Supported Devices

Evaluation Software
FREE CD-ROM
Product Updates
Product Version

Product Information
Why buy tools from Keil
How to place an order
Get more information
Get Prices/Quotes

Version Numbers
Sales Channels
Y2K Readiness

CA51 Compiler Kit

The Compiler Kit for the 8051 microcontroller family (part number CA51) supports all 8051 derivatives from companies like Analog Devices, Atmel, Cypress, Dallas Semiconductor, Infineon, Intel, OKI, Philips, SMC, Synopsis, TDK, Temic, and Triscend.

The CA51 Compiler Kit does not include the µVision2 Debugger. The Debugger is included in the DK51 Developer's Kit and the PK51 Professional Developer's Kit.

The following components are included in the CA51 Compiler Kit.

- µVision2 IDE
- A51 Macro Assembler
- C51 ANSI C Compiler
- BL51 Code Banking Linker/Locator

Internet addresses:

[1] SDCC:
http://sdcc.sourceforge.net

[2] READS51:
www.rigelcorp.com/reads51.htm

[3] Pascal51:
www.eedevl.com/files.html
Pascal51, Prepro51:
www.geocities.com/SiliconValley/Campus/9592/

[4] Embedded Pascal:
http://users.iafrica.com/r/ra/rainier/p8052.htm

[5] Keil C51 evaluation tools:
www.keil.com/demo/

[6] Franklin Development Tools:
www.fsinc.com/devtools/Default.htm

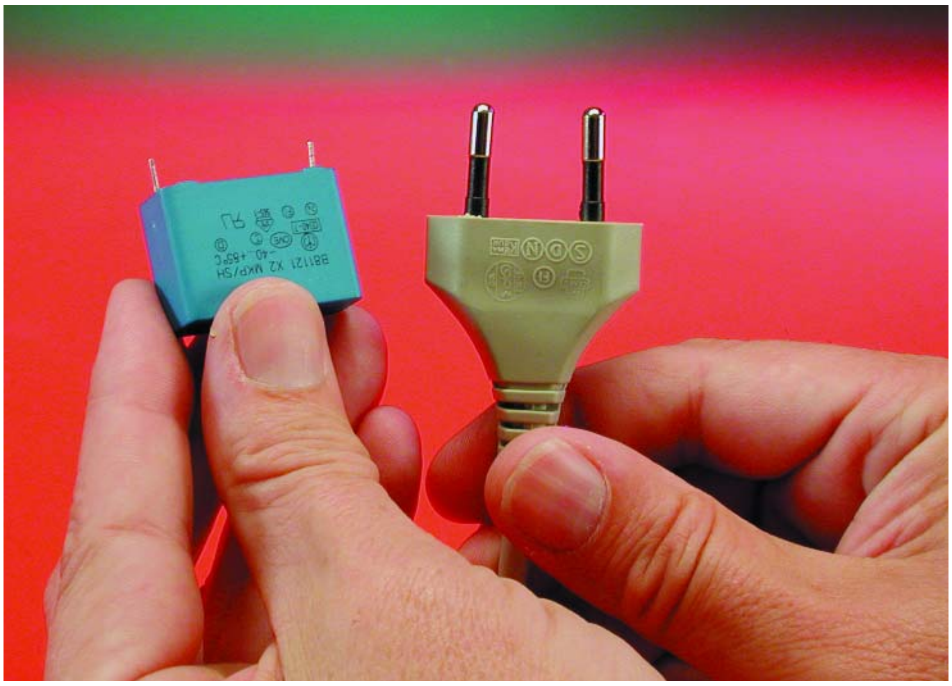
[7] Raisonance 8051-compiler:
www.raisonance.com/download/ index.php

Capacitor as AC Voltage Dropper

transformerless power supply

By S. van Rooij

Some designs, because of lack of space or for some other reason, do not use a power supply transformer. Instead, an 'AC voltage dropper' in the shape of a capacitor is fitted. On more than one occasion readers have asked us the details of how this works and how the value should be calculated.



If the power supply for (part of) a circuit requires only a few milliamps from the mains the well-known 'transformerless' solution is frequently used. Here, a capacitor is used as an 'AC voltage dropper'. This method is not actually used all that often, because the advantage of the smaller size is offset by the disadvantage of reduced electrical safety. Observe that the power supply is coupled directly to the mains and this is not really that elegant a solution, of course.

Nonetheless, an 'AC-voltage dropping resistor' can sometimes be a convenient solution. One of the more recent occasions where we used such a transformerless power supply in *Elektor Electronics*, is the power-on delay for the Crescendo audio power amplifier. **Figure 1** shows the schematic for this design once more.

The exact details regarding the operation of this circuit are not relevant right now, but note that the 24-V relay Re1 is powered from the mains without the use of a transformer.

Capacitor C1 functions in this circuit as the AC-voltage-dropping-resistor. Resistors R1, R2 and R3 may obscure the picture somewhat, but they are of secondary importance in this instance. R1 and R2 make sure that C1 discharges when the mains power is switched off, and R3 limits the inrush current. The necessary voltage drop from 230 V to 24 V is achieved by C1 alone.

How does it work?

We cast our memory back to a bit of elementary theory. To illustrate how this AC voltage dropper works, a simple schematic has been drawn in

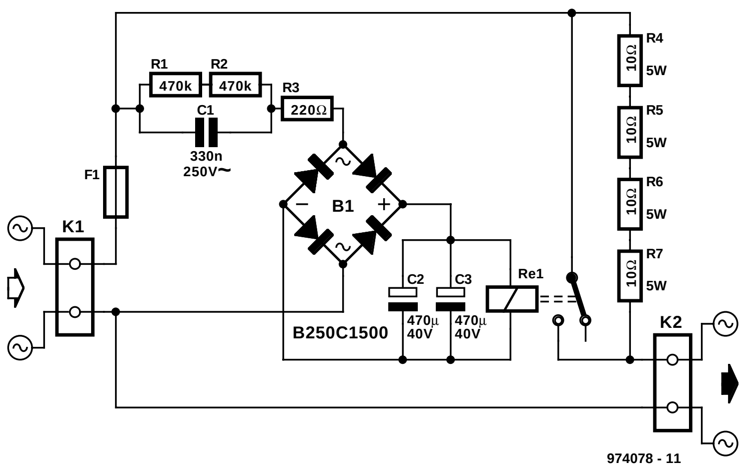


Figure 1. In this circuit, Re1 is powered directly from the mains by making use of 'AC voltage dropper' C1.

Figure 2. The whole circuit consists of a battery, two pushbuttons, a capacitor and a moving coil instrument.

If you activate button S1, the capacitor will charge and a current will flow, for a short period, through meter M. The needle will deflect briefly and then return to zero. If you now release S1 and subsequently push S2, the capacitor will discharge with the result that the needle of the meter will deflect briefly in the opposite direction.

We now take one more step and replace the battery with an AC voltage source. Because the polarity of the voltage is changing continuously, when S1 is held down, the capacitor will be repeatedly charged, discharged, charged in the opposite direction, discharged and so on. The moving coil instrument M will therefore indicate a constant current. This current though, will continuously change direction at the same rate as the AC frequency. If the moving coil instrument is replaced with an AC ammeter (a multimeter in AC current mode) this will then indicate a constant value. From this we can conclude that capacitor C appears to behave just like a resistor. This apparent resistance to AC current is called the **capacitive reactance** of the capacitor. This reactance is frequency dependent and can be calculated using the formula

$X_C = 1 / (2 \pi f C)$ [Ω]

In this formula, *C* is the capacitance of the capacitor in Farad and *f* the frequency in Hertz.

Example

How do we calculate such an AC voltage dropper in practice?

For this purpose, let us briefly return to the circuit of **Figure 1**. The initial requirement is that the relay requires 24 V and therefore there has to be a voltage drop of some 200 V across capacitor C1. It is also necessary, of course, that we know the magnitude of the current that flows through the relay coil; for this type of relay this amounts to 20 mA.

Ohm's Law now permits the apparent resistance of the capacitor to be calculated: approximately 10 kΩ. Note that an actual resistor of this value would dissipate about 4 watts.

Once this is known, we can calculate the value of the capacitor with a rearranged version of the above mentioned formula:

$C = 1 / (2 \pi f X_C)$

We substitute for *f* the mains frequency of 50 Hz and end up with $1 / (2 \times 3.14 \times 50 \times 10^4) = 0.318$ µF. **Figure 1** shows that a value of 330 nF has been selected, which is very close to the theoretical one.

Of course, the capacitor that will be used here has to be rated for at least 250 V **alternating** current, while mains applications also require that an X2-class capacitor be used.

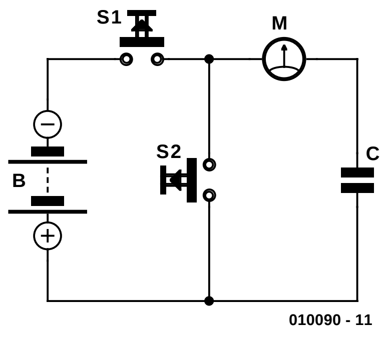


Figure 2. The operating principle can be easily demonstrated with the aid of this test circuit.

Limitations?

We already mentioned that the AC voltage dropper is used mainly when the current is limited to a few mA. Is there some reason or other why this method may not, or cannot, be used with larger currents?

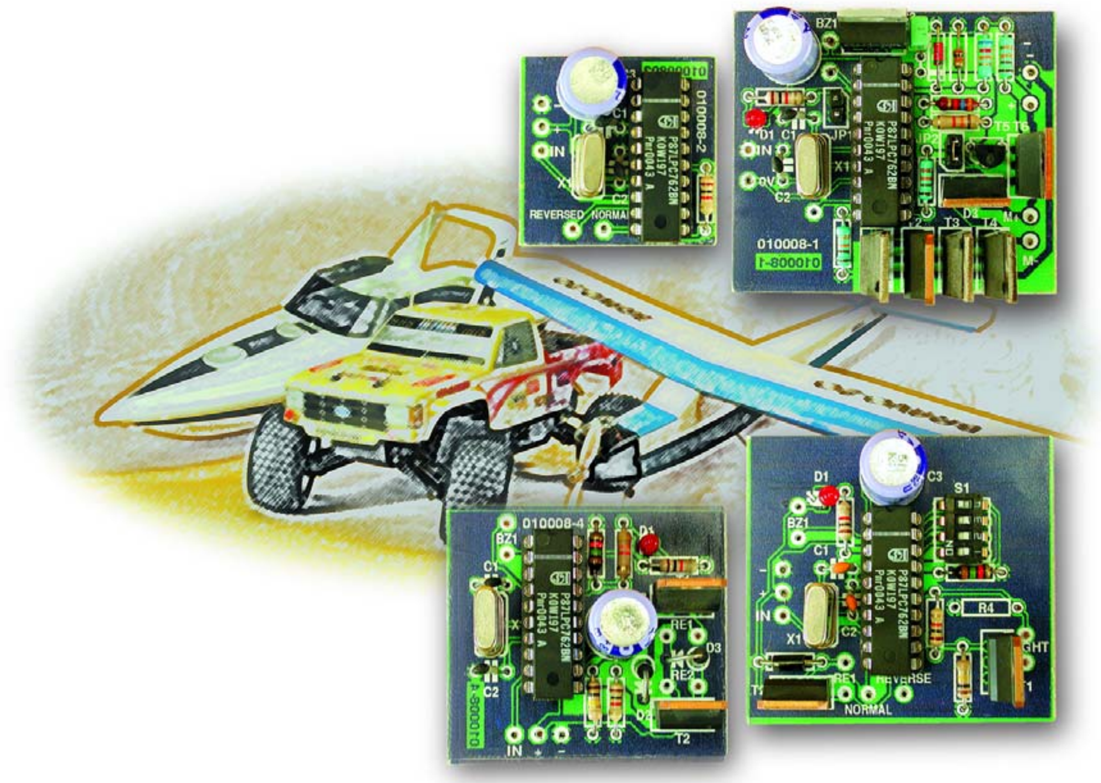
In principle, no. Naturally, the voltage dropping capacitor will work equally well with larger currents as it does with small ones. However, the capacitor that will be required will keep growing in size as the current increases, and at some point it becomes more sensible to use a transformer instead. This transition point occurs at around 100 mA in practice. An X2 class capacitor with a rating of 1.5 µF/250 V~ is nearly the same size as a mains transformer rated 2×9 V/100 mA. At still larger currents, the space required for a high voltage, bipolar, foil capacitor would exceed that of a suitable transformer. Taking electrical safety into consideration as well, it obviously becomes preferable to use a transformer instead.

(010090-1)

Multi-purpose IC for Modellers (2)

part 2: two printed circuit boards and nine functions

The multi-purpose IC for modellers, offering a total of fourteen different functions, really comes into its own with the two printed circuit boards presented here. This board provides the speed control, soft start, motor voltage monitoring, BEC, anti-collision light, fail-safe, hot glow, servo balancing and go-slow functions.



In the previous article in this series the multi-purpose IC for modellers was presented with two application printed circuit boards. In this second and final instalment we present two further circuit boards. In total these four circuit boards offer fourteen different functions, all using the same IC, a Philips 87LPC762 microcontroller specially programmed for this project. The two new application circuit boards are the 'motor control' and the 'go-slow/hot glow' boards. An overview of the four circuit boards and the functions they offer is given in **Table 1**.

Motor voltage monitoring

In remote-controlled aeroplanes and vehicles, batteries are normally used both for drive and simultaneously, via a BEC system (Battery Eliminator Circuit), to supply the receiver. Since the batteries should not be allowed to discharge too far, and because the BEC requires a certain minimum input voltage, it is sensible to monitor the battery voltage and take suitable action when a critical level is reached.

The system must be designed so that these actions are clear to the user and so that they do not jeopardise subsequent operation of the model. As a permanent optical

Table I

Printed circuit board and function overview

Printed circuit board 010008-1, 'speed controller', Elektor Electronics, March 2002:
Speed controller, soft start, BEC, model finder, receiver voltage monitor, motor voltage monitor

Printed circuit board 010008-2, 'servo reverse', Elektor Electronics, February 2002:
Servo reverse, Y-cable replacement, fail-safe, servo filter

Printed circuit board 010008-3, "hot glow/go-slow", Elektor Electronics, March 2002:
Servo reverse, Y-cable replacement, fail-safe, servo filter, anti-collision light, go-slow, aileron balancing, hot glow, receiver voltage monitor, model finder

Printed circuit board 010008-4, "2-channel switch", Elektor Electronics, February 2002:
Fail-safe, receiver voltage monitor, model finder, 2-channel switch

indication a LED is connected to port pin P1.6. Taking into consideration that the motor voltage monitor function will only be used in conjunction with the soft-start or speed control functions of the circuit in **Figure 1**, we can also cause the motor to stutter: this will provide a indication clear to any modeller. This state can be permanently switched off by briefly switching the motor off and then back on with a nudge on the control stick. This ensures that a safe landing can be made using full motor power.

The above actions are taken when the microcontroller detects that the voltage is too low ten times in a row, measured at intervals of 20 ms. If the voltage monitoring function is not required, resistor R2 at port pin P0.2 should not be fitted. Zener diode D2 protects the microcontroller input from voltages greater than the power supply voltage. If R2 is not fitted, the zener diode **must** be fitted to the circuit board (**Figure 2**). Note also that the Zener diode must be a 0.25 W type.

The required values for the resistors to obtain a desired threshold voltage U_S can be calculated using the formula for a voltage divider. The internal reference, to which the voltage on port pin P0.2 is compared, is 1.23 V. If, for example, R3 is set at 10 kΩ, we have for R2:

$$R2 = 10\ \text{k}\Omega * (U_S - 1.23\ \text{V}) / 1.23\ \text{V}$$

For example: 8 cells, final voltage 8.8 V:
$R2 = 10\ \text{k}\Omega * (8.8\ \text{V} - 1.23\ \text{V}) / 1.23\ \text{V}$
$R2 = 61.54\ \text{k}\Omega$ (preferred value: 62 kΩ)

BEC system

A BEC (Battery Eliminator Circuit) is a system

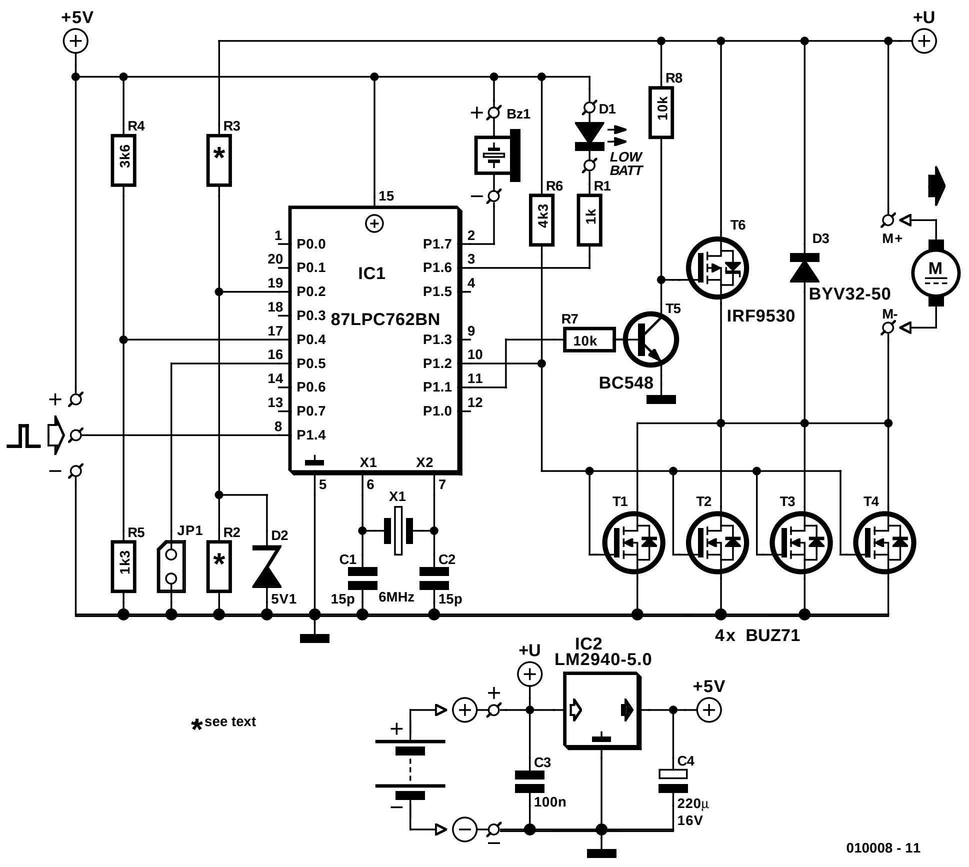


Figure 1. Circuit diagram of the motor control printed circuit board.

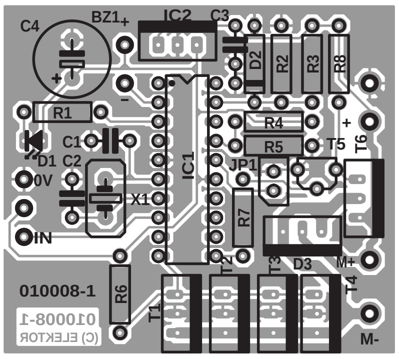
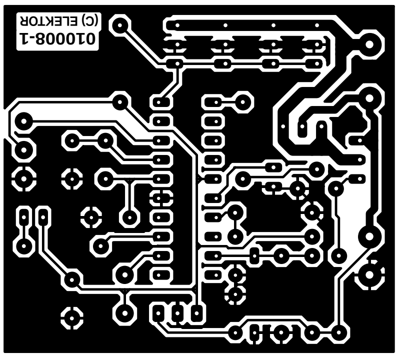



Figure 2. The motor control printed circuit board.

COMPONENTS LIST

PCB 010008-1
(Motor controller, Fig. 2)

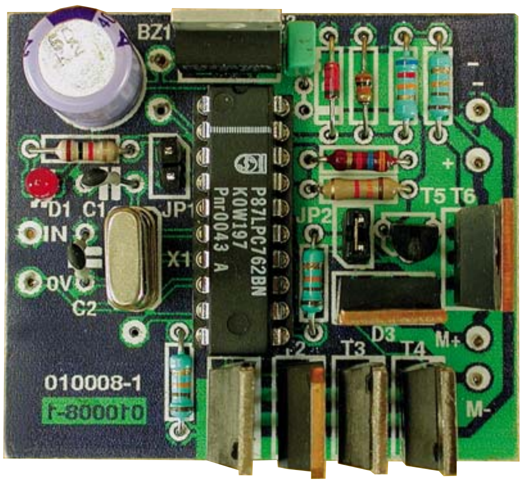
Resistors:
R1 = 1kΩ
R2,R3 = see text
R4 = 3kΩ6
R5 = 1kΩ3
R6 = 3kΩ6
R7 = 10kΩ
R8 = 10kΩ

Capacitors:
C1,C2 = 15pF
C3 = 100nF
C4 = 220µF 16V radial

Semiconductors:
D1 = LED, red, low current
D2 = zener diode 5V1
D3 = BYV32-50
IC1 = 87LPC762BN (programmed, order code **010008-41**)*
IC2 = LM2940-5.0
T1-T4 = BUZ71/IRLZ34N
T5 = BC548
T6 = IRF9530

Miscellaneous:
Bz1 = DC buzzer, 5V
X1 = 6MHz quartz crystal
2-way pinheader with jumper
PCB, order code **010008-1***

* For programmed microcontrollers, PCBs and the project software disk, please refer to the Readers Services pages elsewhere in this issue or the publishers' website www.elektor-electronics.co.uk



Prototype board (slight differences with final version).

which replaces the receiver battery with a supply derived from the main battery. The BEC in the circuit of Figure 1 consists of a low-dropout regulator IC2 which can provide a regulated 5 V output for the receiver from a suitable main battery. If this function is not required, IC2 need not be fitted to the printed circuit board (Figure 2) and the power supply for the circuit wired directly to the point marked '+U'.

If IC2 is fitted, there is a power loss which depends on the difference between the main battery voltage and the BEC output voltage, and on the BEC output current:

$$P = (U_{Bat} - U_{BEC})\, I_{BEC}.$$

This power loss is entirely converted into heat. The greater the voltage difference and the greater the current, the greater the power loss and hence the heat produced. This can lead to a significant temperature rise in the IC if no provision is made to dissipate the heat, via a heatsink, for example. A heatsink is recommended for the voltage regulator to dissipate the excess heat if high-power servos are used. In contrast to previous BEC designs employing the 7805 voltage regulator, the low drop-out voltage of the LM2940 regulator used here allows operation down to an input voltage of 6 V. The chief technical characteristics of the LM2940 are as follows:

- 1 V regulator drop-out voltage
- Maximum output current 1 A at T_J=25 °C
- Reverse-polarity protection
- Maximum input voltage 26 V

Speed control

Modern speed controllers for model aircraft are small, light and powerful and driven by a microcontroller. The control algorithm implemented in the multi-purpose IC allows highly efficient speed control of DC motors at currents up to approximately 40 A using pulse width modulation. The motor current is switched by power MOSFETs T1 to T4, which share the load and which switch at a frequency of about 2 kHz. Flywheel diode D3 protects the transistors from voltage spikes as the motor is switched off.

With the motor is turned off the turning of the propeller in the airstream can adversely affect the gliding characteristics of a model. For this reason a braking function is implemented, using T6 to short-circuit the motor. This brake can be disabled by putting the control stick into the fully on position with the transmitter switched on but the receiver switched off, and then turning on the receiver. Helicopter pilots should be aware of this function, which allows the rotor to turn freely. A built-in safety function ensures that the motor does not switch on instantly if the control stick is not initially in the off position: before the motor will start the stick must first be moved to the off position. The range of accepted pulse widths from the receiver is from 1.2 ms to 1.9 ms.

If the battery voltage monitoring function is used and if the voltage goes below the preset minimum value, the motor begins to stutter rhythmically. This stuttering alerts the model's pilot to the low voltage condition. The stuttering can be stopped by briefly switching the motor off and then on again using the control stick.

Since the received pulse width is measured to microsecond accuracy, the resolution of the speed controller and soft start systems is also one microsecond. This gives very fine control, which many other controllers with coarser PWM quantisation cannot match.

Soft start

The soft start switch is constructed in a similar way to the speed controller. The only difference is that that the motor is automatically ramped up to full speed over a period of 2 s. The switching threshold is set when the multi-purpose IC is switched on. The pulse width measured at that point, plus an extra 400 µs (which in practice corresponds to a control stick movement of approximately half its range), sets the value above which the motor will be started. This safety function reliably prevents the motor inadvertently being started up when the transmitter is turned on with the control stick set to its maximum position. The braking function can also be deactivated here in an analogous way to that described above for the speed controller.

Hot glow

When the soft start or speed control functions are activated in the hot glow/go-slow circuit shown in **Figure 3** (with printed circuit board shown in **Figure 4**), port pin P1.0 provides a signal that switches to indicate when the length of the pulse provided by the receive is less than 1.2 ms. The output is switched on when the control setting is in the given range and switched off when the setting leaves that range. An interesting application of this function is the automatic switching of the glow plug supply in four-stroke engines, which idle rather more smoothly with the glow plugs turned on. If the weight of the model is not critical, it is generally worthwhile fitting a battery for the glow plug supply in the model: this avoids fiddling with the glow plug connector, which can be dangerous if it is near a propeller.

MOSFET T2 can easily handle currents up to a maximum of around 10 A.

Anti-collision light

Light effects are eye-catching, and certainly lighting adds interest to a model. The microcontroller's software generates a signal on port pin P0.7 which can be used for an anti-collision light (ACL). The signal drives the lights via power transistor T1, which can comfortably switch currents of 10 A. The flash rate depends upon the pulse repetition rate of the receiver servo signal outputs, and so can vary slightly from manufacturer to manufacturer. At a frequency of 50 Hz, for example, the lights will be switched on for 0.5 s and then off for 1.5 s.

As already discussed, monitoring the receiver supply voltage is very important for the 'survival' of a model. To indicate when the main or auxiliary battery is (nearly) flat, the flash rate of the anti-collision lights is modified in the supply voltage alarm condition, so that the problem can be detected even from a great distance. The flash rate changes to 4 Hz and stays at that value until the microcontroller is reset and the supply voltage restored to a safe level.

Aileron balancing

'Aileron balancing' refers to unequal upwards and downwards deflection of the ailerons. The objective is to compensate for the so-called negative moment (the tendency of the model to fall out of a turn). Models with a large wingspan and symmetrical deflection of the flaps experience different drag on the two wings which tends to yaw the model about the vertical axis against the direction of the turn. This is normally compensated for by an increased rudder deflection, which, however, wastes power. A better approach is to rebal-

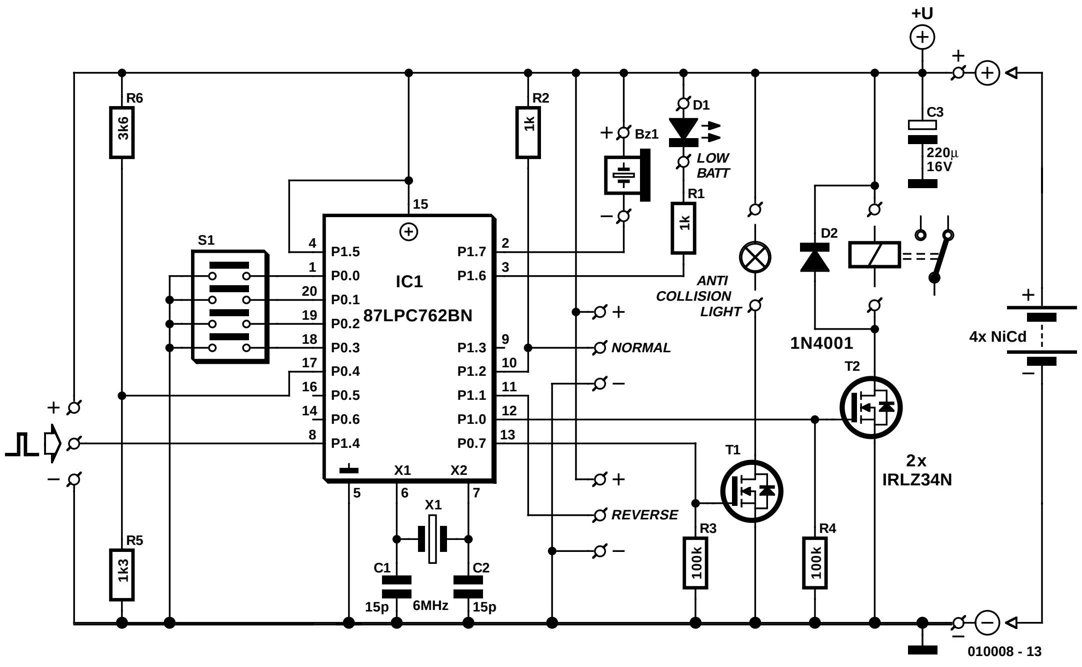
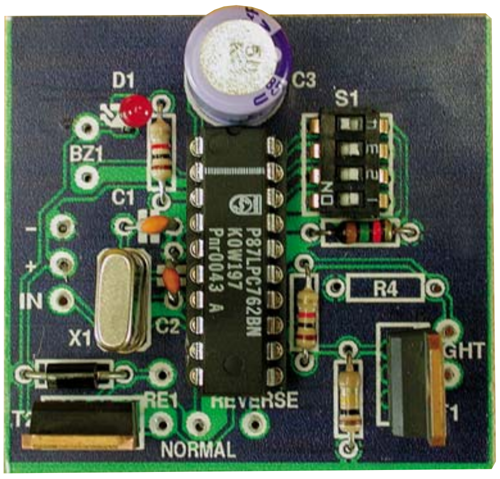


Figure 3. Circuit diagram of the 'hot glow/go-slow' printed circuit board.



Prototype board (slight differences with final version).

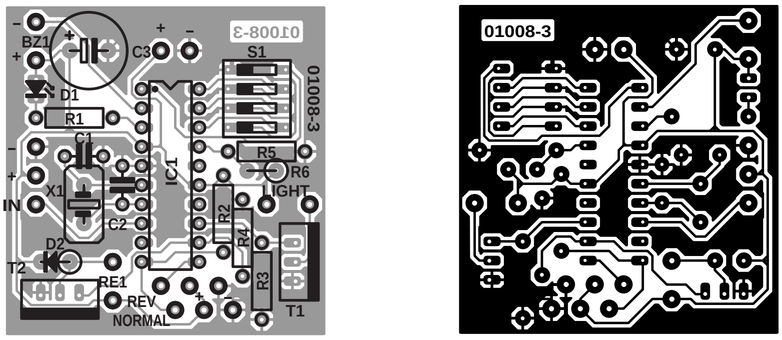


Figure 4. The 'hot glow/go-slow' printed circuit board.

COMPONENTS LIST

PCB 010008-3
(hot glow/go-slow, Fig. 4)

Resistors:
R1,R2 = 1kΩ
R3,R4 = 100kΩ
R5 = 1kΩ3
R6 = 3kΩ6

Capacitors:
C1,C2 = 15pF
C3 = 220µF 16V radial

Semiconductors:
D1 = LED, red, low current
D2 = 1N4001
IC1 = 87LPC762BN (order code **010008-41**)*
T1,T2 = BUZ71/IRLZ34N

Miscellaneous:
Bz1 = DC buzzer, 5 V
S1 = 4-way DIP switch
X1 = 6MHz quartz crystal
PCB, order code **010008-3***

* For programmed microcontrollers, PCBs and the project software disk, please refer to the Readers Services pages elsewhere in this issue or the publishers' website www.elektor-electronics.co.uk

ance the ailerons (**Figure 5**), which generally are positioned either 100 % upwards or 50 % to 70 % downwards. These values will vary depending on the geometry of the model.

The degree of balancing effect is configured via DIP switch S1 on the printed circuit board, the three switches S1-1 to S1-3 setting a value in 8 steps (of 10 %) between 20 % and 90 %. The fourth switch S1-4 selects between the aileron balancing function (switch on) and the go-slow function (switch off). **Table 2** shows the relationship between the switch settings and the aileron difference value obtained.

The balancing effect operates only above the neutral position, that is to say when the pulse width of the signal from the receiver lies in the range 1.5 ms to 2 ms. Both servo outputs are affected by this function, the reverse output being affected when the pulse width is in the range 1.5 ms down to 1 ms. This means that two servos operating in opposite directions can be fitted, even though only one receiver output is used.

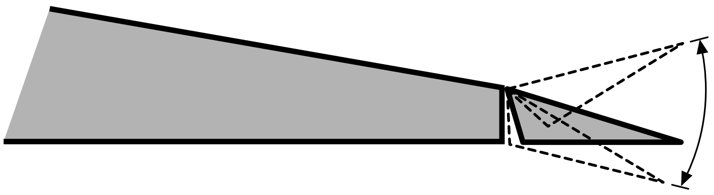


Figure 5. Principle of aileron balancing.

Go-slow

Modern servos are reliable, powerful and fast. On the one hand, speed can often be very desirable, but on the other hand there are times when the servo cannot be made to move slowly enough. For example, moving the landing flaps suddenly to their end position can give rise to an undesirable consequence such as the model rearing up. Also, a very rapidly moving undercarriage mechanism looks unrealistic. Of course, each servo could be smoothly controlled using a separate controller on the transmitter, but in any case any modeller will be happy to have one fewer control to worry about during landing or take-off. It is important to note that the go-slow

Table 2

Relationship between switch settings and aileron balancing value

Switch settings				
1	**2**	**3**	**4**	**Difference**
on	on	on	on	20%
off	on	on	on	30%
on	off	on	on	40%
off	off	on	on	50%
on	on	off	on	60%
off	on	off	on	70%
on	off	off	on	80%
off	off	off	on	90%
off	off	off	off	0% ('go slow' function with zero delay)

Table 3

Relationship between switch settings and delay

Switch settings				delay (s)
1	2	3	4	
on	off	off	off	1
off	on	off	off	2
on	on	off	off	3
off	off	on	off	4
on	off	on	off	5
off	on	on	off	6
on	on	on	off	10
off	off	off	off	zero delay

Note: switch 1 to switch 4 are connected to port pins P0.0 to P0.3
on = switch closed = port pin input sees ground level
off = switch open = port pin pulled up by internal resistor

function does not work with special servos that can only run from one extreme position to the other. These servos cannot be proportionally controlled and therefore also not slowed down.

As far as the servo is concerned the go-slow unit is a black box that sits between the receiver and the servo, measures the incoming pulse, and either increases of decreases the length of the output pulse at a given rate, so that the servo eventually arrives at the position corresponding to the new pulse width at the receiver. The circuit simulates the gradual movement of a control stick at the transmitter, with a speed selectable in seven steps. As well as the applications suggested above, driving the undercarriage mechanism or landing flaps, other possibilities that might lead to interested glances in the direction of your model might include a smoothly-opening cockpit hood or door. The relationship between switch settings and servo speed are given in **Table 3**, where switch S1-4 is responsible for activating the go-slow function (when the switch is in the 'off' position).

The go-slow function also allows two servos to be connected in such a way that they operate in opposite directions but at the same speed. Note that the times given are only indicative values and are dependent on a 25 Hz pulse repetition rate from the receiver. Again, there may be differences here from manufacturer to manufacturer.

Fail-safe

The fail-safe function, which has until now only been available on PCM remote control systems, forces the servos into defined positions in the case of severe interference in the received signal. Whether a model aircraft can carry out a safe landing with the rudder fixed at the mid-position is questionable, but switching off the motor and electrical functions is certainly sensible to minimise the extent of any possible damage. For example, a jammed motor running on the full supply voltage can easily destroy the driver electronics with the excess current. It is also sensible to return the rudder to its mid-position in the case of interference rather than, for example, leaving it at maximum deflection. The fail-safe function is activated when the interference in the received signal is so severe that the microcontroller can no longer decode a valid signal or when the received signal is completely absent (because the model is out of range or because the transmitter has been switched off). The microcontroller then:

1. activates the buzzer;
2. deactivates all switch outputs;
3. in the case of the soft start system and the speed controller, switches off the motor;
4. drives both servo outputs with a pulse width of 1.5 ms.

The fail-safe measures remain in force until a valid received signal is once again available.

MOSFETs

Finally, a few words on the MOSFETs used. Modern examples of this type of transistor feature practically loss-free switching and a low 'on' resistance. It is important to ensure, however, that the gate is always fully driven so that the transistor does not operate in the resistive part of its characteristic. Since only 5 V is available to drive the transistors, so-called 'logic level' types must be used: these allow high currents to be switched without getting unduly hot. The International Rectifier type IRLZ34N suggested is readily available and offers a good price/performance ratio. Its characteristics are as follows:

– I_D:	27 A
– V_{DS}:	55 V
– P_D:	56 W
– R_{DSON}:	0.035 Ω

Modellers who want to squeeze the last drop of performance out of their system might consider using type SUP75N03-04, which, with an R_{DSON} of 0.004 Ω, offer an even lower voltage drop.

(010008-2)

Internet addresses

LM2940:
http://www.national.com/pf/LM/LM2940.html

N-channel MOSFET:
http://www.irf.com/product-info/datasheets/data/irlz34n.pdf

P-channel MOSFET:
http://www.irf.com/product-info/datasheets/data/irf9530.pdf

BYV32-50 :
http://www.mobilesemi.com/us/products/fer.asp

SUP75N03 :
http://www.nessel-elektronik.de/FET_Bauteile/fet_bauteile.html