FREE C PRIMER

elektor electronics

leading the way

MARCH 2006
£3.80
www.elektor-electronics.co.uk

FPGA Module

fresh technology for you, too

Schematics on centrefold!

Practical:

- Cheap Logger
- Travel Charger

- Triumphant March of the 6502
- Development Kits
- Linear Motors

Application Board for R8C/13

03
9 770268 451128

Micros turn Mega

I've always been confident that the Greek prefixes μ (micro) and M (mega) are safely apart at an immense factor of 10^{12} or one billion. The venerable prefixes are established in electronics to the extent that only rank beginners would surmise the existence of microhertz frequencies and megafarad capacitances. Sure, these exist in theory but no-one's ever seen the equipment or capacitor to match.
Although it's not necessary right now to burn your maths textbooks, recently the distance between micro and mega has dwindled and in fact both prefixes can now be said to co-exist happily in two species of electronic component. The main subjects of this month's issue, microcontrollers and FPGAs, execute microinstructions from megabytes of memory, not forgetting nanosecond signal setup times and several kilobits a second worth of data traffic on serial lines.
If you thought that the microcontroller was the next and perhaps final step in an evolutionary process that started with the valve and went by way of the transistor and the integrated circuit, buckle up for FPGAs. To illustrate the scale at which we need to think now, some FPGAs are powerful enough to mimic a vintage 6502 microprocessor including its exact pin functionality, and still have room to spare for 'side activities' like serial interfacing and memory flashing.
Readers tell us that they find FPGAs impressive and even awe-inspiring, but also off putting not just for design complexity but mostly 'solderability'!
Here at *Elektor Electronics* we're not in the least afraid to deal with the latest in FPGA and microcontroller land, witness our Versatile FPGA Module, FPGA Prototyping Board and Renesas R8C/13 Motherboard. Let the professionals do the soldering — in all three cases we supply the boards with all the dreaded SMDs, BGAs and TQFPs and what have you ready fitted, so you can concentrate on the real thing: programming and developing turnkey applications with maximum intelligence and an absolute minimum of components. I wouldn't say wielding a full-blown FPGA or R8C development system is a micro effort at a nano investment but you can be certain of mega results. Glad to help you.

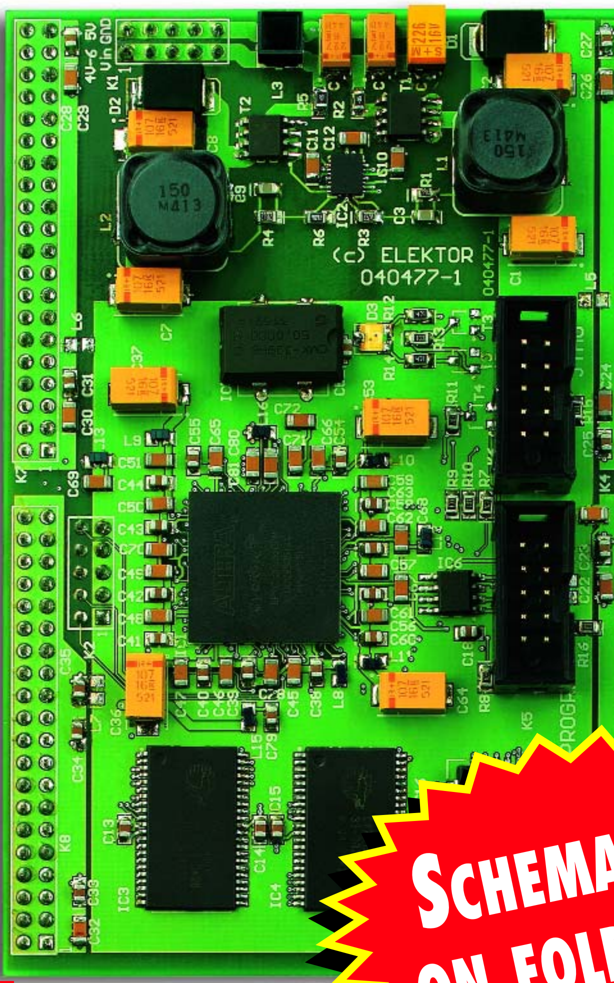
Jan Buiting
Editor

elektor electronics leading the way

16

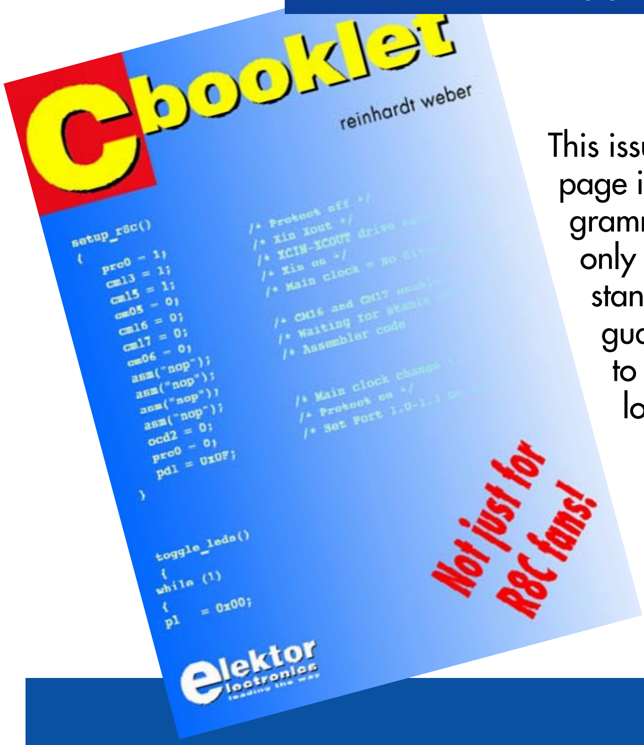
Until recently, FPGAs were practically reserved for specialists in high-tech companies. That's all changed now thanks to low component prices and free design software, so it's high time to devote attention to this technology in Elektor Electronics. The FPGA module discussed in this article forms the core of various upcoming projects in the magazine.

Versatile FPGA Module



SCHEMA
ON FOI

Free C Booklet

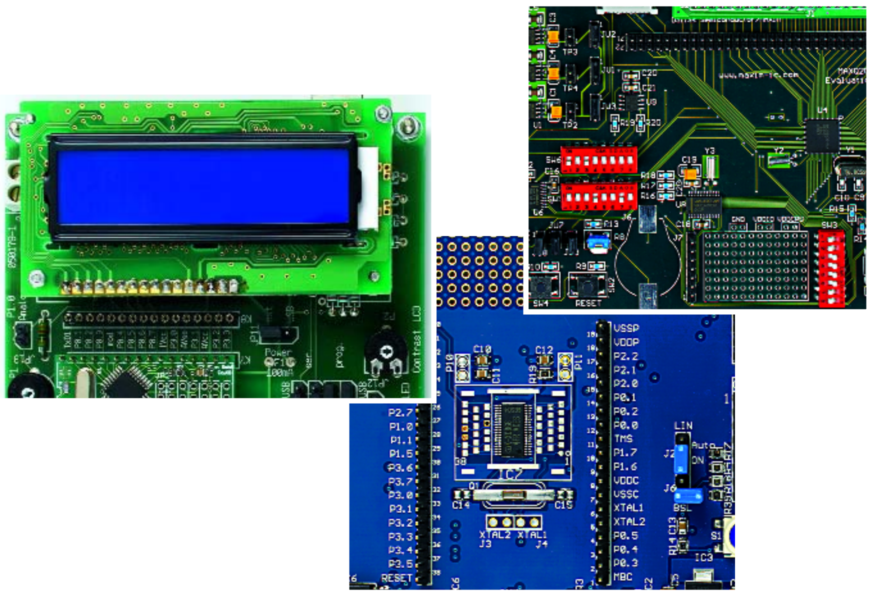


This issue comes with a free 24-page introduction to C programming. The booklet is not only a runway to an industry-standard programming language, but also a prelude to articles on C for our low-cost R8C Starter Kit.

CONTENTS

Volume 32
March 2006
no. 352

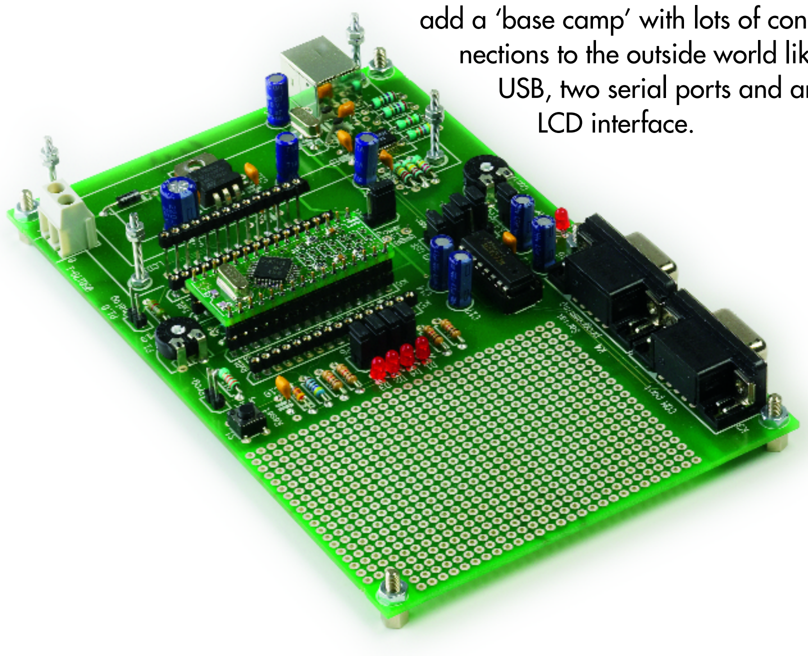
28 Development Kits



We devote attention to interesting but affordable kits where we not only look at kits that are suitable for a commercial environment but which are also very interesting for hobby use.

38 Application Board for R8C/13

Last month we described how you can get started with our under-£10 R8C/13 microcontroller module. The Now we add a 'base camp' with lots of connections to the outside world like USB, two serial ports and an LCD interface.



Volume 32, Number 352, March 2005 ISSN 0268/4519

Elektor Electronics aims at inspiring people to master electronics at any personal level by presenting construction projects and spotting developments in electronics and information technology.

Publishers: Elektor Electronics (Publishing), Regus Brentford, 1000 Great West Road, Brentford TW8 9HH, England. Tel. (+44) (0) 208 261 4509, fax: (+44) (0) 208 261 4447 www.elektor-electronics.co.uk.

The magazine is available from newsagents, bookshops and electronics retail outlets, or on subscription. *Elektor Electronics* is published 11 times a year with a double issue for July & August.

Under the name *Elektor* and *Elektuur*, the magazine is also published in French, German and Dutch. Together with franchised editions the magazine is on circulation in more than 50 countries.

International Editor: Mat Heffels (m.heffels@segment.nl)

Editor: Jan Buiting (editor@elektor-electronics.co.uk)

International editorial staff: Harry Baggen, Thijs Beckers, Ernst Krempelsauer, Jens Nickel, Guy Raedersdorf.

Design staff: Ton Giesberts, Paul Goossens, Luc Lemmens, Karel Walraven

Editorial secretariat: Hedwig Hennekens (secretariaat@segment.nl)

Graphic design / DTP: Ton Gulikers, Giel Dols

Managing Director / Publisher: Paul Snakkers

Marketing: Margriet Debeij (m.debeij@segment.nl)

Subscriptions: Elektor Electronics (Publishing),
Regus Brentford, 1000 Great West Road, Brentford TW8 9HH, England.
Tel. (+44) (0) 208 261 4509, fax: (+44) (0) 208 261 4447
Internet: www.elektor-electronics.co.uk
Email: subscriptions@elektor-electronics.co.uk
Rates and terms are given on the Subscription Order Form

Head Office: Segment b.v. P.O. Box 75 NL-6190-AB Beek The Netherlands
Telephone: (+31) 46 4389444, Fax: (+31) 46 4370161

Distribution: Seymour, 86 Newman Street, London W1P 3LD, England

UK Advertising: Huson International Media, Cambridge House, Gogmore Lane, Chertsey, Surrey KT16 9AP, England.
Telephone: +44 (0)1932 564999, Fax: +44 (0)1932 564998
Email: r.elgar@husonmedia.com
Internet: www.husonmedia.com
Advertising rates and terms available on request.

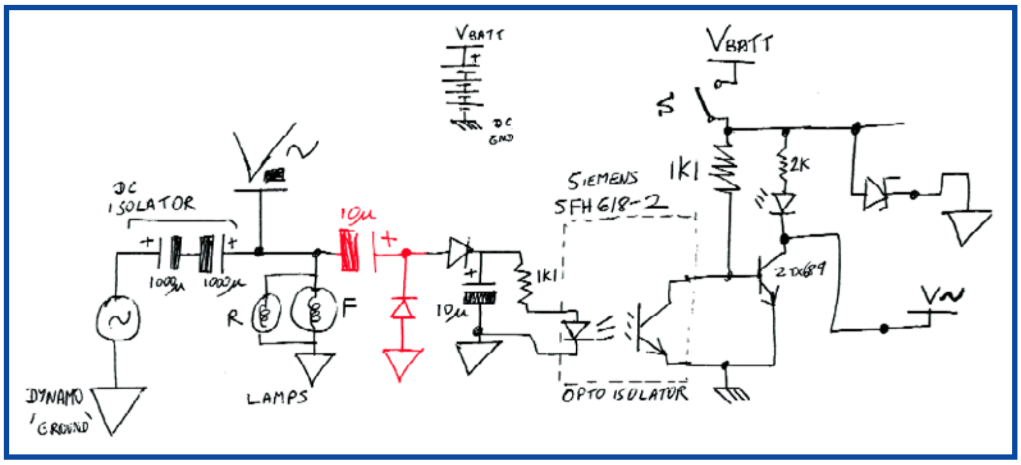
International Advertising: Klaas Caldenhoven, address as Head Office
Email: advertenties@elektuur.nl
Advertising rates and terms available on request.

Retronics on multimeters (3)

Dear Sir — I enjoyed very much Jan Buiting's article on analogue multimeters ('Retronics', December 2005, *Ed.*). I myself own an excellent BBC Goerz Unigor A43 analogue multimeter that I bought at a flea market. It has a DC sensitivity of 100 kohms/volt and its lowest DC current scale is 10 microAmperes.
In addition to the advantages of analogue over digital meters mentioned in the article, I would like to call your attention to another one mentioned by Robert A. Pease in his book *Troubleshooting Analog Circuits*. On page 17 he says:

"Another advantage of analog meters is that they are passive devices: They don't inject noise into your circuit as digital meters can — even battery powered ones. And they have a lot less capacitance to ground".
Mr. Pease is referring to the noise generated inside digital meters by the A/D conversion process, of course.
Congratulations Jan for your interesting article.
Carlos Urbina Pacheco (Mexico)

Thanks for the compliment Carlos, as they say, "it makes a welcome change from being at the wrong end of the customer complaints line". The photograph shows an Unigor 4S meter from my collection.



Battery backup for bike — help please

(Mailbox December 2005 issue refers, *Ed.*)
Dear Editor — I have got my dynamo backup circuit to work: a voltage doubling rectifier solved the problem caused by the half-wave rectifier's series capacitor.

I enclose the corrected schematic. Note that the DC isolator capacitors are mounted separately beside the dynamo, whereas the circuit is mounted at the other end of the bike (near the battery pouch).
The battery is also used for extra lamp(s) such as a turn indicator.
The circuit seems to take 3 mA from battery when switched over to dynamo but even if that's just noise picked up by the meter, I can live with it.
I found the free Linear Technology LTSpice program very useful.

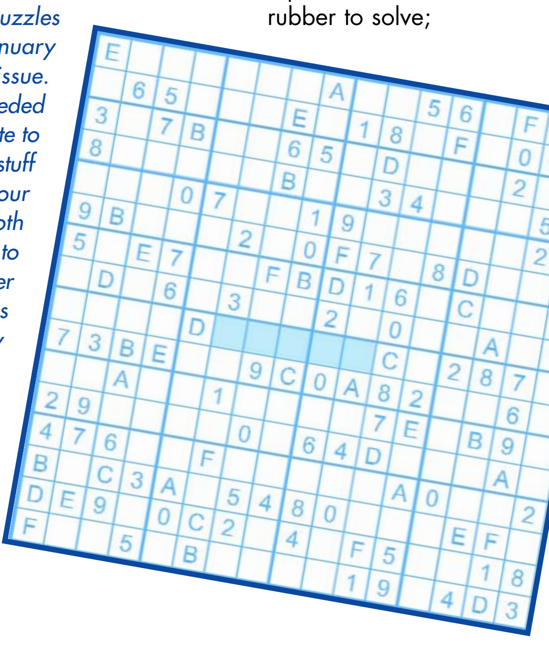
An alternative circuit (simulated with LTSpice) replaced the optoisolator and BJT with a logic-level p-channel MOSFET (protected by a zener), and connected battery ground to dynamo ground.
Unfortunately, Maplin no longer sell p-type MOSFETs so I could not build this circuit.
I have also enclosed a photo of the finished unit. The handlebar bracket is a Minoura battery lamp extension (long) minus the lamp tube.
Alan Bradley (UK)

That's a happy ending then Alan, congratulations on getting your circuit to work. We also thank those readers who have written with suggestions to help Alan find (and solve) the problem. He also informed us that he has further information available on the migration of the circuit to FETs.

Hexadokus

While compiling this month's Mailbox pages, we're inundated with solutions to our first and second 'Hexadoku' puzzles published in the January and February 2006 issue. Apparently, we succeeded in adding a lighter note to high-brow electronics stuff that normally fills our pages. We were both pleased and surprised to see that a good number of correct solutions were sent by family members of our readers. A number of solutions reached us by regular post (thank you all for using our new address). Readers were also quick to spot that our Hexadokus can be downloaded free of charge from our website. Here's some comment we got from you:

"A great opportunity for us lesser gods in electronics to win a prize".
"Can we have a microcontroller version of it please."
"Took me two days to solve but I think I have the solution."
"What a wonderful idea this Hexadoku; just what I would expect from Elektor."
"I just love Sudokus and was hooked the moment my husband showed your hexadecimal variety. It's addictive and crazing. And tough, too."
"The normal 9x9 Sudoku was starting to bore me. Your hex variety got my grey matter milling over numbers again."
"A great puzzle. I was already hooked on Sudokus but yours is even worse!"
"Cost me one pencil and a rubber to solve;

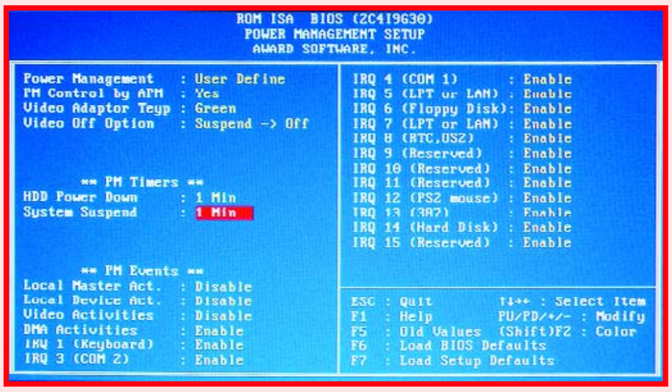


Rejektor

Under this heading we will occasionally publish circuits, ideas and suggestions that did not make it to full publication in this magazine for various reasons (like lack of space).

A power-wise server

ADSL, wideband and fast cable Internet connections spreading very fast across countries in the world, many of you may start to think about using an old PC as a server.
Unfortunately many forget the server's additional power consumption adding considerably to the electricity bill. The average consumption of a Pentium 133 PC without a CRT monitor is between 50 and 100 watts. A Pentium P2-266 consumes even 25% more! A typical CRT monitor will consume 80 watts on average. Money-wise that's about 60 pounds per year if the equipment is left on all the time.
Mr Jeroen Baars from the Netherlands sent us his suggestions for reduced power consumption for small network servers, based on his own experience with a Pentium 133 PC. The best candidates for this kind of server applications are PCs with a CPU running at anywhere between 90 and 233 MHz and having a Socket-7 type motherboard. Although a Socket-5 or a 486 DX/DX2/DX4 may also be suitable, these machines will be found to be sluggish and have somewhat higher power consumption.
To begin with you should check if the PC is still fully functional. Run a thorough test on the hard disk, then make sure it's virus-free. Make notes of the hardware address and IRQ of the network card. Next you can disassemble the PC to prepare it for its future function. Open up the power supply and remove dust and dirt with a small brush. Watch out for residual charge of the electrolytic capacitors.
Connect the fan to 5 V instead of 12 V and check it runs smoothly and reliably on the lower voltage. If not, replace it by one that does.
Replace the CPU cooler by a version without a fan. In most cases the fan can be removed from the heatsink and the cooling will still be sufficient. Replace a high-performance video card you may come across by a standard card with just 1, 2 or 4 Mbytes of memory. If an ISA card is used, make sure the hardware address and IRQs are known (for the software installation). A PCI card will usually stand a higher chance of success provided it is from a reputed supplier. A sound card is not required. A floppy disk or IDE card is only necessary with a 486 machine. The available memory should be 16 Megabytes or more.
Once everything looks fine so far, a cheap energy meter (£15-20 from builders markets) and the following measures will tell you how much energy is saved.
To start with, relocate the CPU speed jumpers to select a setting of about 75 MHz. With some motherboards and CPUs, even lower speeds may be set up. Although apparently even more energy-wise, in practice you'll find that 25 to 30 MHz is the minimum you can get away with. In most cases however, you'll find that "the slower the more economical". There are exceptions however. When testing a stripped-down 486DX machine no difference was noticed when the clock speed was reduced from 50 down to 25 MHz. You may also want to experiment a little with the CPU core voltage jumpers, making sure you never select a voltage higher than the original setting. If you happen to have a small stock of video, network and (if applicable) HD/floppy disk cards, it's very well worth the time and effort to swap these around for a bit as one card may consume far more power than the other.
If you assemble the PC without the help of an energy meter, be sure to use a PCI video card.
After all this modding around, the PC will typically consume 25 to 30 watts. Of course, there will be exceptions and tough cases but then these are easily recognised from boiling hot chips and ditto heatsinks.
In principle, a floppy disk is sufficient for a small website, router or printer server. You will need a hard disk however if want to run a large website with photos on it and FTP access. The hard disk may, of course, be replaced with an IDE Flash drive: very power-wise, quiet and perfectly working!
The BIOS settings will usually require some changes. The BIOS will be specific to the motherboard used, so some research will be needed to find and understand the settings that matter. Enable the 'halt on errors' option in the Standard CMOS Setup menu. Then go to the Power Management menu and change the settings to those indicated in the screendump. Next, click on 'Save & Exit'. At that point you are ready to proceed with the installation of an operating system, for example, Freedos or Freesco. Both have a "dot org " web address.



```
                    ROM ISA  BIOS (2C4I9630)
                    POWER MANAGEMENT SETUP
                     AWARD SOFTWARE, INC.

Power Management   : User Define        IRQ 4 (COM 1)        : Enable
PM Control by APM  : Yes                IRQ 5 (LPT or LAN)   : Enable
Video Adaptor Teyp : Green              IRQ 6 (Floppy Disk)  : Enable
Video Off Option   : Suspend -> Off     IRQ 7 (LPT or LAN)   : Enable
                                        IRQ 8 (RTC,OSZ)      : Enable
                                        IRQ 9 (Reserved)     : Enable
                                        IRQ 10 (Reserved)    : Enable
      ** PM Timers **                   IRQ 11 (Reserved)    : Enable
HDD Power Down      : 1 Min             IRQ 12 (PS2 mouse)   : Enable
System Suspend      : 1 Min             IRQ 13 (3A7)         : Enable
                                        IRQ 14 (Hard Disk)   : Enable
                                        IRQ 15 (Reserved)    : Enable

      ** PM Events **
Local Master Act.   : Disable           ESC : Quit          ↑↓→← : Select Item
Local Device Act.   : Disable           F1  : Help          PU/PD/+/- : Modify
Video Activities    : Disable           F5  : Old Values    (Shift)F2 : Color
DMA Activities      : Enable            F6  : Load BIOS Defaults
IRQ 1 (Keyboard)    : Enable            F7  : Load Setup Defaults
IRQ 3 (COM 2)       : Enable
```

next one, please!"
"Took a bit longer than the decimal ones!"
"Nice puzzle; blood, sweat and beer!"
"Could not put it down and got off the wrong subway station."
"A most enjoyable distraction, I will look forward to more of these puzzles in the future."

Visual Basic Course

Dear Elektor people — I was thrilled to see the free Visual Basic course booklet with the January 2006 issue. A fantastic initiative, thank you Mr Petros Kronis and Elektor for your efforts!
When I wanted to download Visual Basic Express edition 2005 I got a message saying the product required Windows 2000. Unfortunately, my PC has Windows 98. Does this mean I am unable to proceed? I was unable to find a 'minimum requirements' list in your booklet.
Jose Qvik (Netherlands)

The big advantage of the latest version of Visual Basic is that it's free. Most programs discussed in the booklet will also run using older versions of Visual Basic, but these, alas, are not free. Maybe you should start looking for an older VB version someone is willing to sell at a low price.

PIC programming

Dear Editor and lab staff — I am looking for a handbook

Corrections & Updates

SC Analyser 2005

April 2005, p. 34, 030451-1

Several readers have reported difficulties with the configuration bits to set up in the PIC. The confusion is owing to poorly defined standards at Microchip. Because all status bits are at '1' with a new processor, there are options where '1' means 'enabled', while for other options it means 'disabled'. To add to the confusion, some programmer systems (including the one currently in use at Elektor labs) employ the notation 'on' and 'off'.
The essential options are the oscillator, watchdog timer, low-voltage programming and in-circuit debugger. The following list provides the desired state for each option, where '0' = unprogrammed and '1' = programmed.

Options:

bit 13, CP:	Flash Program Memory Code Protection bit; **1**
bit 11, DEBUG:	In-Circuit Debugger Mode bit; **1**
bit 10-9, WRT1:	WRT0 Flash Program Memory Write Enable bits; **11**
bit 8, CPD:	Data EEPROM Memory Code Protection bit; **1**
bit 7, LVP:	Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit; **1**
bit 6, BOREN:	Brown-out Reset Enable bit; **1**
bit 3, PWRTEN:	Power-up Timer Enable bit; **0**
bit 2, WDTEN:	Watchdog Timer Enable bit; **0**
bit 1-0, FOSC1:	FOSC0: Oscillator Selection bits; **11**

DDS RF Signal Generator

October 2003, p. 14, 020299-1

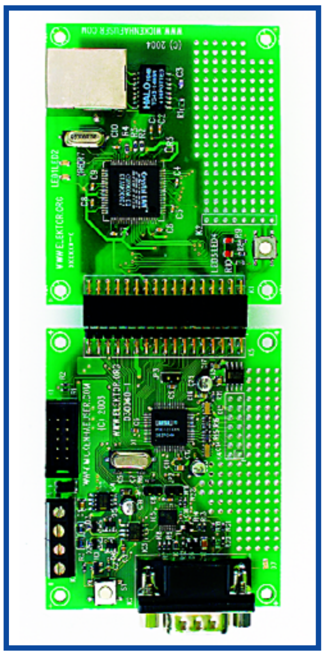
Pin 6 of IC3 is erroneously connected to the +9 V supply rail. The relevant IC input is however designed to handle logic levels with a maximum High level of +5 V. Although the IC will not take damage from the +9 V level applied, it is recommended to cut the relevant connection and wire pin 6 to the +5 V supply rail.

or a course covering PIC microcontroller programming. Several questions in forums failed to produce relevant answers, so I am now turning directly to you. Do you have any relevant books?
Chris (UK)

All books we have available are faithfully listed on our website, see SHOP → Books & CD-ROMs. Unfortunately there's not yet a book that matches your exact requirements. However, we found one on the Internet at www.mikroelektronika.co.yu/english/product/books/PICbook/picbook.htm *and a series of tutorials at* www.mstracey.btinternet.co.uk/pictutorial/picmain.htm

MSC1210 programming tweak

Dear Editor — I am happy to report further progress with my Elektor MSC1210 microcontroller system (*Precision Measurement Central*, July August 2003 and *Micro Webserver*, July/August 2004, *Ed.*). I happened to discover that replacing the instruction printf("%f ....) somewhere in the program by printf("%u ....) reduces the amount of code generated by about 20%. Apparently, floating point output is very byte consuming. The 20% extra space I gained in this way allowed me to extend the program to my personal requirements.
Patrick (Belgium)



Thanks for letting us know Patrick, I'm sure other MSC1210 users will find this useful information.

Four steps to LEDs on the mains

Dear Jan — I really liked the extra-large instalment of *Design Tips* in the January 2006 issue. Unfortunately, in a number of cases the component references in the texts with *Four Steps to LEDs on the Mains* are not consistent with the schematics you have printed. I'm happy to say that despite the discrepancies I did not have problems understanding the operation of the circuits. Many thanks for an interesting article.
Walther (UK)

Unfortunately, the component numbering in the drawings was changed from the author's hand drawn originals we used to produce the texts. Also, LED D10 in the third drawing is shown the wrong way around. Given the simplicity of the circuits discussed we trust the errors do not detract too much from the educational value.

MailBox Terms

– Publication of reader's correspondence is at the discretion of the Editor.
– Viewpoints expressed by correspondents are not necessarily those of the Editor or Publisher.
– Correspondence may be translated or edited for length, clarity and style.
– When replying to Mailbox correspondence, please quote Issue number.
– Please send your MailBox correspondence to:
editor@elektor-electronics.co.uk or
Elektor Electronics, The Editor, 1000 Great West Road, Brentford TW8 9HH, England.

British computing power helps bring Were-Rabbit to life

Sight Systems has helped bring the latest Aardman Animations film, "Wallace & Gromit: The Curse of the Were-Rabbit", to the screen with their high-tech custom computer systems. Sight Systems helped build a custom-developed computer called the Aardcase that helped Nick Park and his team to speed up the process of bringing the plasticine hero's, Wallace and Gromit, to life as well as creating the huge, mysterious veg-ravaging beast: the Were-Rabbit!

The Aardcase is connected to the film cameras, capturing the images from up to 30 cameras directly to hard disc. It allows the animators to check the minute changes made from one frame the next in real time. Although many people assume that the technology behind blockbuster movies is associated with large Hollywood companies, Sight Systems has provided some technology for several productions, including the last Aardman film, "Chicken Run".

The Aardcase is a single board based PC with a passive backplane in a ruggedised 6U 19" rack mount case. Aardman demanded a highly robust product that could be easily transported – including being wheeled over the cobbled paths near their office – so Sight Systems designed a custom vibration-resistant solution to ensure that the computer would stand up to this rough treatment. The system includes a video capture PCI card, an optical drive for storing captured film, and two hard drives. The two drives separate the video, which is stored on a high-performance SCSI drive, from the operating system and software that is held on a cost-effective ATA drive.

The custom case routes the video connections and other I/O to a rear panel designed for use by animators with little or no knowledge of PCs. In addition to the video inputs, the system also provides a 12V supply to power the cameras. Installed software includes the Perception Video Recorder from DPS as well as a custom Video Assist and Digital Video Frame store developed by Aardman that helps the animators plan character movements on screen, and allows them to check the frame currently being shot with those already completed to ensure continuity.

"We have provided many systems for the film and broadcast industry, which is only one of the applications that use our products," said Phil Walters, MD of Sight Systems. "Our long relationship with Aardman shows that we can meet the needs of the most demanding customers over an extended period of time."

Wallace & Gromit: The Curse of the Were-Rabbit is showing now at local cinemas, featuring the voices of Helena Bonham-Carter and Ralph Fiennes as well as Peter Sallis, who has been the voice of Wallace since the first short.

More information is available at www.sightsystems.co.uk.

(065079-4)

In-vehicle video recorder

Reliance Motor Services (RMS) has announced their success with deView`s Mobile Digital Video Recorder System (DV-M8), manufactured by Security Manufacturing Limited (SML).

Reliance Motor Services are renowned for their success and are the 2005 winners of the "Bus Operator of the Year" award at the RouteONE Excellence Awards." due to their quality and commitment to their customers and staff. RMS operates throughout the Vale of York. RMS contacted Sewell of Leeds nearly 6 months ago to find out what would best suit their requirements for CCTV for double deck buses, which were being added to their fleet. Sewell of Leeds arranged to demonstrate the deView DV-M8 and gave them the option of a range of deView cameras and driver screens to meet their needs.

The DV-M8 can be installed into any vehicle type and you can choose what you require to view internally or externally on the vehicle. Camera installation is quick with well-designed camera mounting making set up quick and simple. The DV-M8 system is easy to add or integrate options such as reversing camera systems, as well as being compact and having the ability to bolt directly to the vehicle without compromising reliability makes installation a breeze.

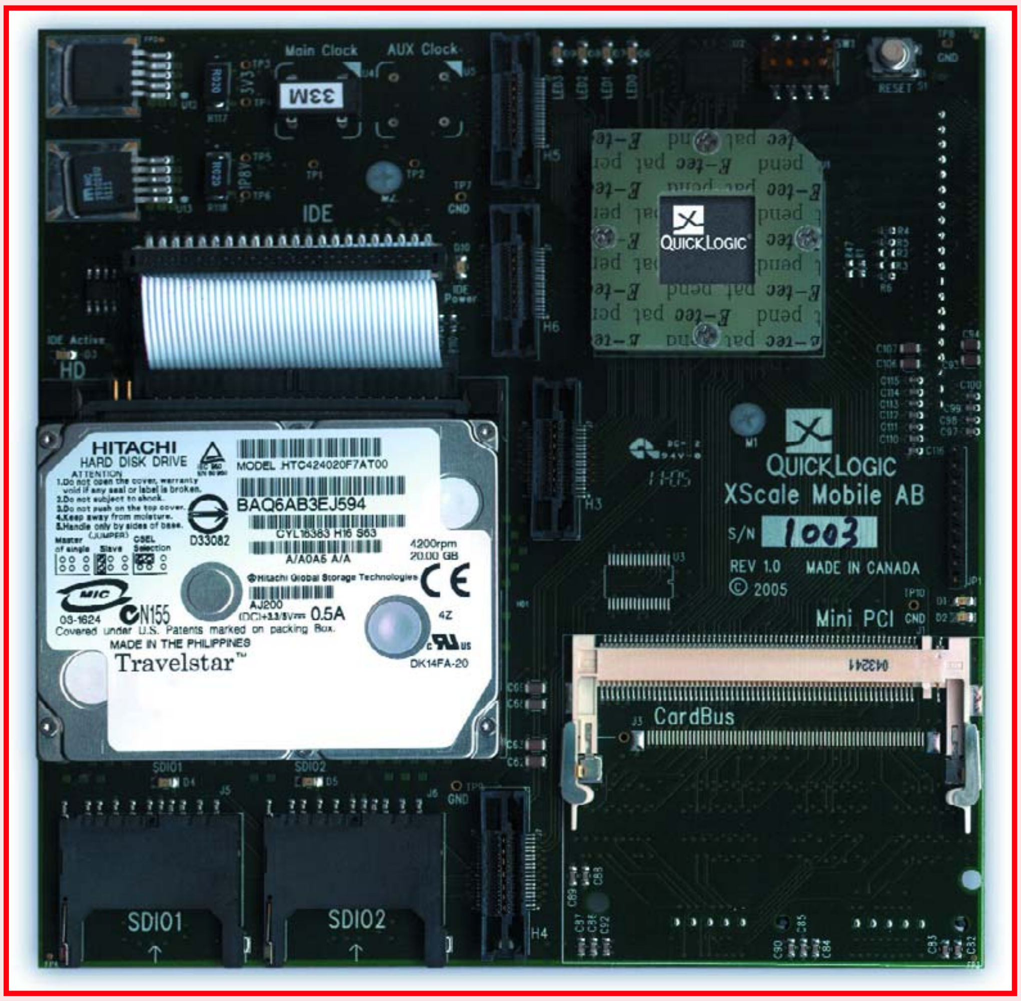
Full details of all deView products can be found on their website at www.deview.com and for more information regarding the DV-M8, please contact Gerry Burns, Special Project Sales.

(065079-6)

FPGA-based mobile applications board

QuickLogic Europe has released a low-power FPGA development tool for Intel's PXA27x based Processor Developer's Kit - the 'Mainstone' DVK - enabling rapid adoption of emerging technologies that are not natively supported. The new Mobile Application Board (MAB) can provide seamless connectivity between XScale processors and peripheral interfaces such as Mini PCI, CardBus, SDIO and IDE.

QuickLogic's FPGA-based MAB enables the development and performance verification of design ideas for a variety of peripheral interfaces such as Wi-Fi, HDD (Hard Disk Drive), 10/100/1G Ethernet, MPEG devices, SD memory, and many more. It connects directly into the VLIO connector of the DVK, which allows designers to make system measurements and architecture trade-offs including power consumption, performance and cost.
MAB allows electronic design teams start software development prior to receiving final hardware. This ability to design hardware and software in parallel can provide product developers with a critical competitive advantage, by significantly reducing time spent in development of popular portable products such as GPS, smart phones, portable media players, and portable industrial systems.
The QuickLogic Mobile Application Board is available immediately. Complete solution packs are available for both Wi-Fi and HDD connectivity. They include the Mobile Application Board, an Eclipse II QL8325 device in a 484-ball BGA package, reference design files and software drivers for Windows CE, Windows Mobile, and Linux.
In addition to this solution for QuickLogic's Eclipse II family of FPGAs, the company will release a MAB based on PolarPro, the new ultra low power FPGA, at the end of Q1, 2006.



For more information, access:
www.quicklogic.com/mab
QuickLogic Europe,
15 London Street,
Chertsey KT16 8AP, UK.
Tel. +44 1932 579011.
www.quicklogic.com

(065079-5)

Advertisement

Your price conscious PCB supplier

EURO CIRCUITS

Online price calculation
Online ordering
Online order tracking
Online 24/24H and 7/7D

Interested? Contact us: +32 15 28 16 30
E-mail: euro@eurocircuits.com
www.eurocircuits.com

Verified
- pooling for standard boards
- up to 6 layers
- from 1 to 1000 pieces
- from 4 working days onwards

à la carte
- technology at the right price
- up to 8 layers
- from 1 to 1000 pieces
- from 3 working days onwards

On demand
- your board, our challenge
- up to 20 layers
- from 1 piece onwards
- from 2 working days onwards

Three-phase PWM controller

International Rectifier announced a three-phase PWM control IC with integrated drivers for DC-DC converters. The IR3094MPbF enables up to 40% circuit size reduction when used with International Rectifier DirectFET™ MOSFETs. The smaller footprint makes the IR3094 ideal for space-constrained applications like DDR memory in rack servers and point-of-load (POL) modules used in high-density data systems.
A typical 80A, three-phase synchronous buck solution would require as many as 13 devices including one control and three drivers in addition to three MOSFETs per phase. A chip set made with the IR3094 plus IRF6637 and IRF6678 DirectFET MOSFETs reduces the silicon component count to seven devices for the same output current. Three pairs of these DirectFET MOSFETs can be placed directly next to the IR3094, creating a solution that minimizes printed circuit board trace parasitics and enables optimum switching performance.
The IR3094 is designed for applications requiring a 0.85 V to 5.1 V output. The new IC is housed in a compact 7mm x 7mm MLPQ package, and features 3A gate drive capability, a 1% accurate reference voltage, adaptive voltage positioning and programmable switching frequency up to 540kHz. The IR3094 provides system protection with programmable soft start, hiccup over-current protection, over-voltage protection, and a “power good” indicator.
The IRF6678, an ideal synchronous MOSFET, features a very low typical device on-resistance of 1.7mOhm at 10VGS and 2.3mOhm at 4.5VGS. The IRF6637 is best suited as a control MOSFET, with very low

Miller charge of only 4nC and typical device on-resistance of 5.7 mOhm at 10VGS and 8.2mOhm at 4.5VGS. Both MOSFETs are housed in the medium can DirectFET package, occupying the same board area as a SO-8 with only a 0.7mm profile, and improved thermal performance through double-sided cooling.

(065079-3)

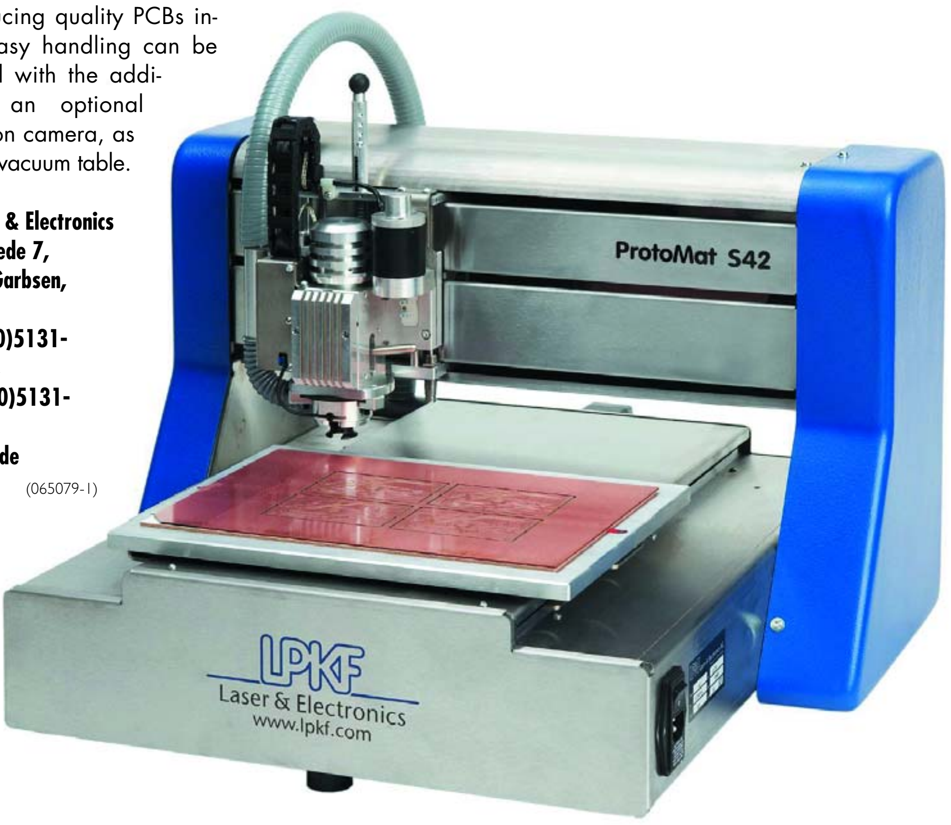
www.irf.com

LPKF ProtoMat S42 PCB Plotter

LPKF Laser & Electronics AG presents the LPKF ProtoMat® S42 circuit board plotter, a new entry-level circuit board plotter for in-house rapid PCB prototyping. This compact system provides precision and performance for quickly and easily milling and drilling single-sided and double-sided circuit board prototypes in a single day.
The LPKF ProtoMat® S42 is an excellent tool for colleges and technical institutions, allowing students and instructors to produce printed circuit boards that are production quality and chemical free, immediately in the classroom environment. Low consumable costs, instant turnaround, and no need for external vendors encourages more practical exercises and experiments in the classroom or laboratory.
The ProtoMat® S42’s 42,000 RPM spindle motor makes it an excellent entry-level performer for producing quality PCBs in-house. Easy handling can be increased with the addition of an optional recognition camera, as well as a vacuum table.

LPKF Laser & Electronics AG, Osteriede 7, D-30827 Garbsen, Germany. Tel.: +49 (0)5131-7095-324, Fax: +49 (0)5131-7095-90. www.lpkf.de

(065079-1)

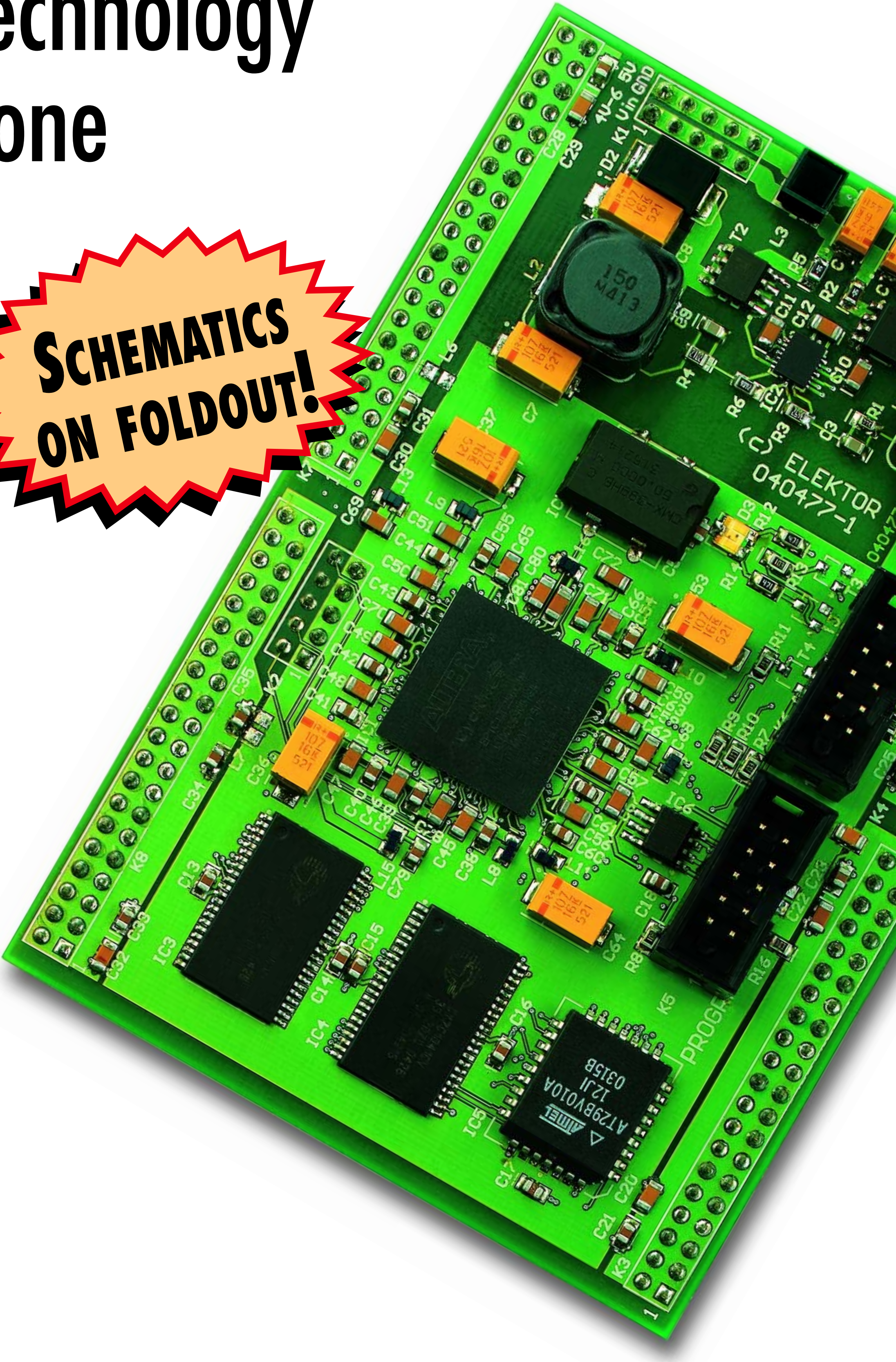


Versatile FPGA M

Modern technology for everyone

Paul Goossens

SCHEMATICS ON FOLDOUT!



FPGAs have established a firm position in the modern electronics designer's toolkit. Until recently, these 'super components' were practically reserved for specialists in high-tech companies. That's all changed now thanks to low component prices and free design software, so it's high time to devote attention to this technology in *Elektor Electronics*. The FPGA module discussed in this article forms the core of various upcoming projects in the magazine.

Module

If you want to use an FPGA in your design, there are several things you can't do without, including a programming interface and configuration memory. We thought it would be a good idea to incorporate the standard items normally associated with an FPGA into a single electronics module, in order to avoid the need for everyone to reinvent the wheel each time. This module can then be used as a 'digital core' for various circuits. A major benefit of this approach is that it allows designers to concentrate on specific applications without having to worry about the soundness of the FPGA element.

Getting started is the hard part

For those of you who haven't used FPGAs before, this module and the associated prototyping board described elsewhere in this issue form an ideal starting point for learning about FPGAs.

It's impossible to avoid using SMDs in a circuit such as this. To make things even worse, we decided to use an FPGA in a BGA package for this circuit, which means DIY soldering with a normal soldering iron is simply impossible. Soldering the other member of the family (in a PQFP package) is also very difficult. However, using these 'difficult' components allows the overall dimensions of the circuit board to be kept reasonably compact.

Fortunately, you don't have to worry about assembling the board, because the FPGA module is available from *Elektor Electronics* with most of the components pre-assembled. The only components you have to solder by hand are the connectors.

Besides this module, you will need a programming interface that sits between the PC and the FPGA board. Naturally, we've also developed a design for this interface.

Technical features

- **Altera Cyclone FPGA**
- **12,060 logic elements**
- **4 MB configuration memory**
- **8 MB user SRAM**
- **1 MB user flash RAM**
- **on-board 50-MHz clock**
- **JTAG/programming interface**
- **Byteblaster compatible**
- **80 user I/O lines**
- **dedicated clock signals**
- **indicator LED**
- **built-in switch-mode power supply**
- **small multilayer PCB (110 x 77 mm)**
- **supplied ready to use**

It takes more than just an FPGA to make a circuit

As already mentioned, you'll need some peripheral electronics to enable you to use an FPGA in a practical circuit. One of the most important elements beside the FPGA is the configuration memory. In contrast to most FPGAs, the configuration memory retains its data when the supply voltage is switched off.

Each time the unit is switched on, the FPGA must be configured again before it can fulfil its intended function. Fortunately, FPGA manufacturers have developed special memory ICs that can configure FPGA ICs automatically when power is switched on. Our circuit includes one of these memory ICs. Easy programming of the FPGA and the configuration memory during the development phase is also highly desirable. That's why a programming interface (using JTAG) also forms part of the standard peripheral circuitry.

Another quite important part of such a circuit is the power supply. The internal operating voltage of most such ICs is quite low (1.5 V in our case). The power supply must be able to handle short current spikes without problems. In addition, a different voltage is used for the inputs and outputs. Here we decided to use 3.3 V. The supply for this voltage must also be able to deliver fairly heavy currents and remain stable under heavy loads.

Our circuit also includes an oscillator, SRAM and flash memory. They can be used freely by the application.

All the individual elements of the circuit can be easily identified in the schematic drawing.

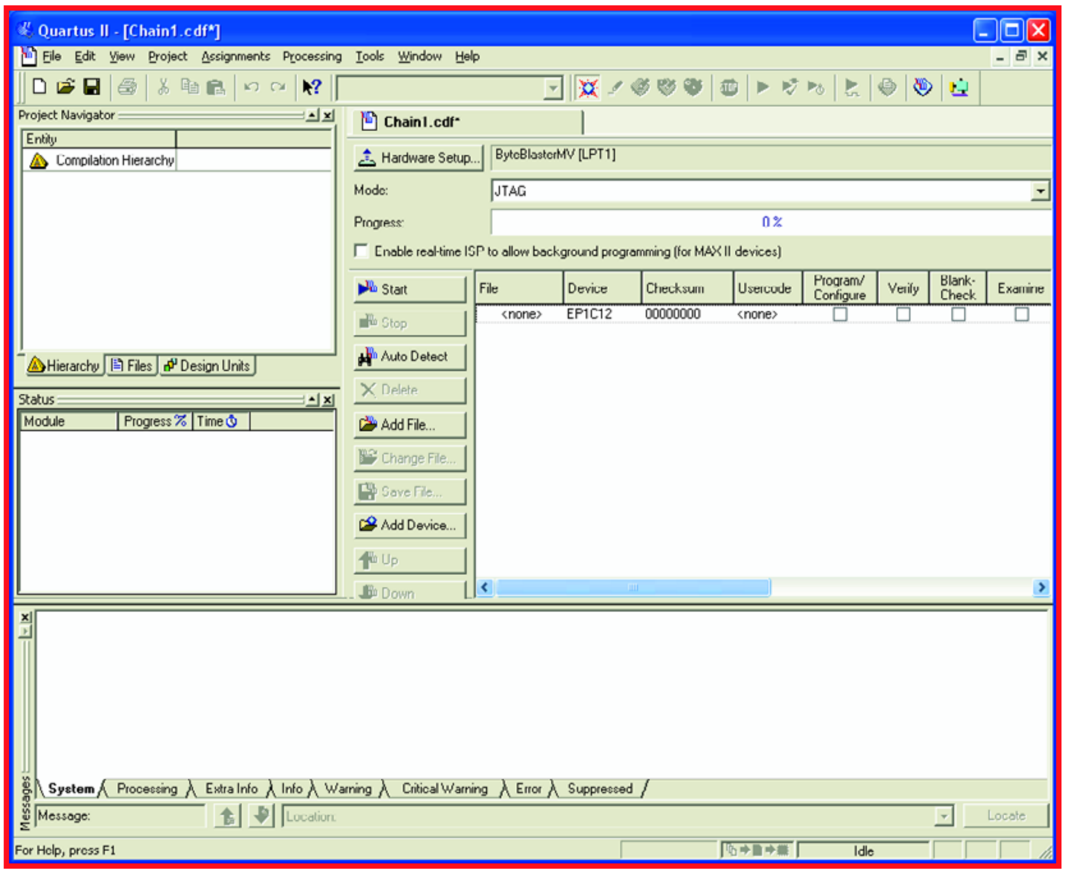
The design

Due to the size of the schematic drawing of the FPGA module, we decided to print it on a foldout which may be found in the centre of the magazine.

Let's start at the beginning with the power supply. It is built around a TPS75003 **(IC2)** and can work with an input voltage in the range of 4.5–6.5 V. This IC is specially designed for use in FPGA circuits. It contains two switch-mode power supply circuits and one series regulator. The latter is not used in our circuit; only the two switch-mode circuits are used.

A step-down regulator that provides a 3.3-V supply voltage is implemented using T1, D1, L1, C1 and associated components. IC2 periodically drives FET T1 into conduction to allow a current to flow from the input voltage via T1, L1 and C1. This causes the current through the inductor to increase, which in turn causes capacitor C1 to be charged. When IC2 switches off the FET, the current flowing in L1 cannot stop immediately, so it continues to flow through C1 and D1. R1, R3 and C3 provide feedback so IC2 can determine whether it has to supply more power or less power.

The values of these components have been chosen such that IC2 tries to maintain the output voltage at 3.3 V. Resistor R2 is a sense resistor. IC2 will limit the current if the voltage across



Hardware Setup... ByteBlasterMV [LPT1]

Mode: JTAG

Progress: 0 %

Enable real-time ISP to allow background programming (for MAX II devices)

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine
<none>	EP1C12	00000000	<none>	☐	☐	☐	☐

Figure 1. The programming mode of the Quartus program.

this resistor rises above 0.1 V. The value used here (50 mΩ) results in a current limit threshold of 2 A.
A similar circuit is built around T2, with only the values of the feedback resistors being different. They are chosen to yield an output voltage of 1.5 V.
The electrolytic capacitors at the input buffer the input voltage. They also prevent fast current spikes from leaving the board. That prevents generation of interference that could impair the operation of nearby equipment.

The FPGA is powered from these supply voltages, but not directly. The supply lines for the two sections of the FPGA are first fed through ferrite beads to block high-frequency radiation. The supply voltages are also buffered close to the IC by several additional capacitors. That may appear to be a bit of overkill, but fairly large switching currents at high frequencies can flow through the FPGA, depending on the final circuit configuration. It is always a good idea to keep the paths of such currents as short as possible. Otherwise the circuit can easily generate too much undesirable interference.

And now for the digital part

Now that we've looked after the supply voltages, we can turn to the digital portion of the circuit. The first thing the FPGA needs is a configuration memory. That memory is present in the circuit in the form of **IC6,** which has been designed to work with the FPGA. When the MSEL signals from the FPGA are at ground level, the IC expects to find a configuration memory (such as the EPCDS4) connected to it. The FPGA will then independently control the memory IC and read data from it in order to configure itself based on the data. That makes the process very easy in actual use.
A programming connector **(K5)** is provided for programming **IC6.** The pin assignments of this connector have been chosen to make it compatible with the Altera 'Byteblaster' programming interface.
Connector **K6** is used for in-circuit programming of the FPGA from a PC. That makes it unnecessary to first program a design into the configuration memory. This connector can be used to quickly test a design without using the configuration memory. Here again, the signals are arranged to be compatible with the Byteblaster interface.
The circuitry around T3–T5 forms an indicator circuit that clearly displays the status of the FPGA using bicolour LED D3. Any error states that may occur are also indicate visibly by D3.

An abundance of I/O

As this circuit is intended to be used for all sorts of applications, it has an abundance of I/O pins. They are accessible via connectors **K3, K4, K7, K8** and **K2.** These connectors fit standard pin headers with 0.1" pin spacing. That avoids the difficulties of using SMD connectors on the motherboard and thus makes it quite easy to connect the board to your own hardware.
Connectors **K3, K4, K7** and **K8** present an impressive total of 80 I/O pins of the FPGA to the outside world. The I/O lines are all routed to one side of each connector. The supply voltage (3.3 V) and ground are available on the other side on alternating pins. You can use the supply and ground pins to power your own circuit if it doesn't require too much current. This supply voltage is also filtered on the module by a ferrite bead and two capacitors.
Connector **K2** provides a special connection. Besides ground lines, it has several signals that are specifically intended to be used as clock signals. Pin 10 is an input that can be used to supply an external clock signal to the FPGA. The remaining even-numbered pins are all connected to the outputs of the internal PLLs of the FPGA.
Of course, it's possible to provide an external clock signal on any other desired input of the FPGA, but these pins are specifically intended to be used for that purpose.

The extras

Many FPGA-based designs require memory. Although there is memory available in the FPGA, it's not enough for many types of applications. In light of the fact that a 'softcore' processor is often used, a flash memory can be quite handy for storing the firmware of the processor. That makes it possible to use larger amounts of code in application designs without taking up additional space in the FPGA. The flash memory takes the form of **IC5** in the schematic diagram.

A bit of extra RAM can also be desirable in some cases. Besides being useful if one or more softcore processors are used in the FPGA, additional RAM can also be welcome for various types of signal processing.

The RAM memory is provided by **IC3** and **IC4.** These ICs have a capacity of 4 Mbits and are configured as 256K 16-bit words. Signals $\overline{\text{BHE}}$ and $\overline{\text{BLE}}$ can be used as two separate $\overline{\text{CE}}$ lines, with $\overline{\text{BHE}}$ being the chip-enable line for the upper byte (D8–D15) of the data lines and BLE the chip-enable line for the lower byte (D0–D7). That means this memory can also be used as a 512-KB memory with 8-bit data.

IC3 is connected directly to the FPGA, while **IC4** shares its data and address busses with the flash memory **(IC5)**. The busses are shared to ensure that enough I/O pins of the FPGA are kept free for the user connectors. However, that means the FPGA cannot read or write the RAM and flash memory at the same time. That doesn't form a major problem in practice, especially when softcore processors are used.

That completes the description of the main parts of the circuit.

We already mentioned that the circuit board for the module is provided nearly fully assembled. The only thing you have to do is to solder the eight provided connectors to the board. Connectors K2, K3, K4, K7 and K8 are fitted on the bottom of the board. The other three connectors are fitted on the top of the board (on the component side). Make sure pin 1 of each connecter is in the right position, which is marked by a '1' on the component overlay.

Programming interface

This module by itself is only a starting point. The FPGA and/or configuration memory must be programmed before they can be used. That requires a programming interface. The schematic diagram of the programming interface is also shown on the foldout. It connects the FPGA or configuration memory to the parallel port of a PC. The programming interface is compatible with the Altera Byteblaster interface and can be used with free Altera software. The circuit is quite straightforward.

Everything is powered by 3.3 V taken from the FPGA module. The two ICs are used to convert the +3.3-V signals from the FPGA to +5-V signals for the PC and vice versa. The 100-Ω resistors prevent the +5-V signals from raising the supply voltage above the allowable level.

The interface is connected to the parallel port of a PC via connector **K3. K2** is a box header that is connected to the FPGA board by a 10-way flat cable with a matching plug at each end.

If you just want to program the FPGA in order to test something, this cable must be connected to **K6** (JTAG) on the FPGA module. If you want to program the module so it can operate on its own, this cable must be connected to **K5** (PROGRAM) on the FPGA module.

Software

A project of this sort naturally includes corresponding software. The manufacturer of the FPGA we use here, Altera, has a very nice package of support software for its FPGAs. The free version of the software is called 'Quartus Web Edition' and can be downloaded from the manufacturer's website (www.altera.com). It's a quite sizeable download (240 MB at the time of writing this article), but you can also request a free CD-ROM on the Altera website.

After installing the software, which by the way goes without a hitch, you will need a licence. You can also obtain this via the Altera website, and it is valid for six months. At the end of the six months you can request a new licence. For the sceptics among our readers, the manufacturer has assured us that this software will remain free in the future.

There's not enough room here to show you everything you can do with this software, and it's equally impossible to present a full course in VHDL here. If you're new to using FPGAs, you should have a look at the accompanying FPGA Prototyping Board described elsewhere in this issue.

Testing

Although this circuit does not have any real input or output capabilities, it's still possible to make a simple test.

After installing the software, it's best to switch off your PC for the next steps. Start by connecting the programming interface to the parallel port of your PC. Next, use the supplied 10-way flat cable to connect the interface to the FPGA module. You should connect the flat cable to connector **K6 (JTAG)** on the FPGA module. Now switch on the PC and connect a power source to the FPGA (caution: maximum voltage 6.5 V).

Start up Quartus and then click on the programming icon in the menu bar, or click on the Tools menu and select Programmer (the result will be the same in either case). The programming window will be opened. Click on the Hardware Setup button, and then click on Add Hardware in the new window. Select 'Byteblaster' and specify the printer port you connected to the programming interface (usually LPT1:).

If you then click on the Autodetect button in the main window, the program will automatically detect your FPGA (EP1C12). That way you can be sure the FPGA is working and the supply voltage is present.

(040477-1)

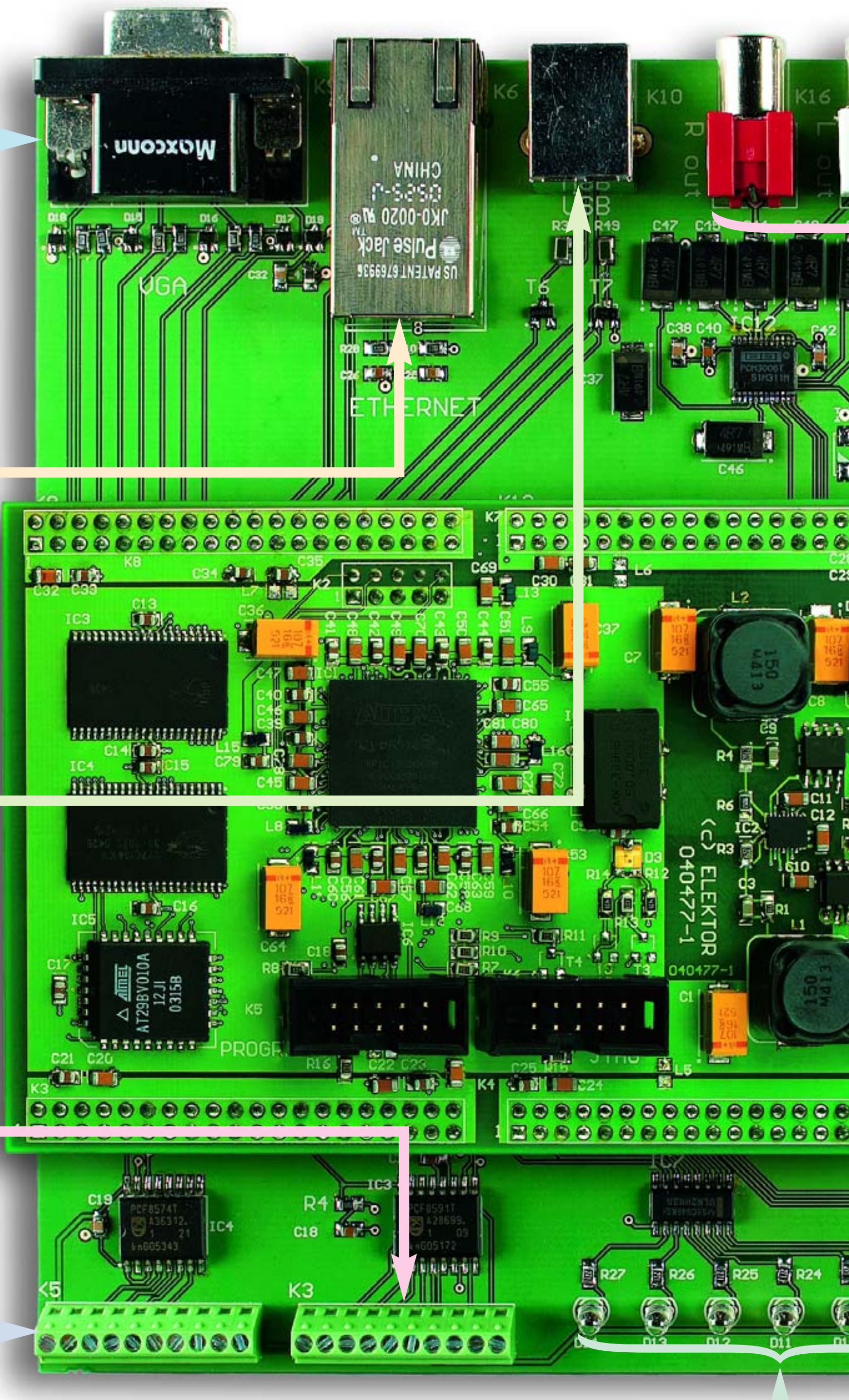
The Versatile FPGA module can be ordered from
Elektor Electronics Readers Services
under order number 040477-91.

Included items

- FPGA module (ready to use)
- programming interface (ready to use)
- programming cable
- interconnect cable (between PC and programming interface)
- 8 pinheaders (supplied separately)

FPGA Prototyping

Paul Goossens



VGA

A genuine VGA output that you can use to display text and imagery on a PC monitor. It's all done with only a few standard components.

See page 24.

Ethernet

Your link to the Internet. The associated IC provides the coupling between the analogue and digital portions.

See page 24.

USB

This USB interface makes communication with your PC fast and easy. And it takes only five components, including the connector!

See page 24.

Analogue I/O

Four analogue inputs and one analogue output. In addition to all the digital artillery on the board, an analogue interface is naturally indispensable. This I/O port has a resolution of 8 bits, which is enough for most applications.

See page 24.

Digital I/O

This interface allows you to connect your own circuitry to the prototyping board.

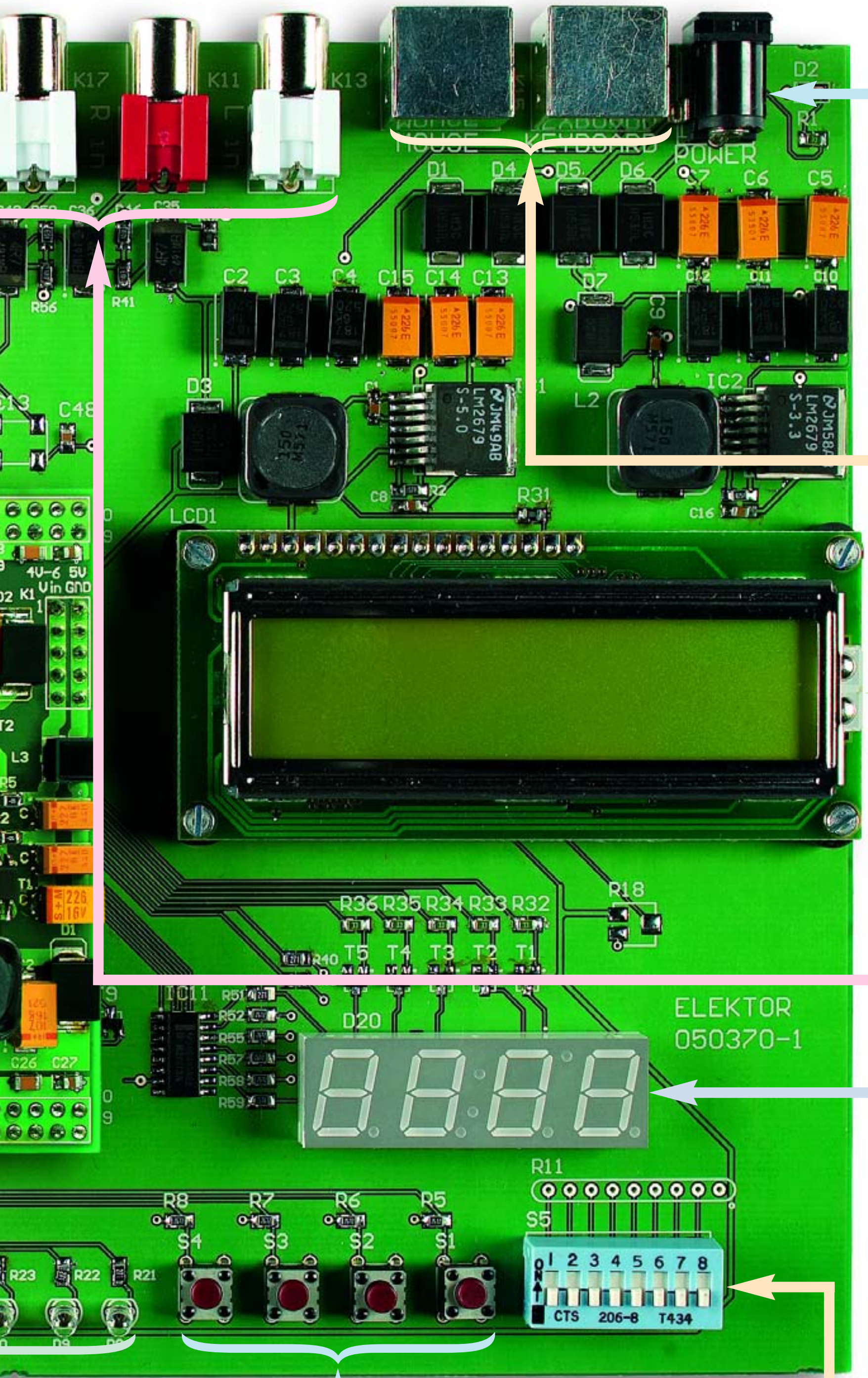
See page 24.

LEDs

You can use the LEDs for a visible indication of the status of various components.

See page 24.

Board FPGA comes to life



Power supply

The power supply is pretty tolerant. As long as the input voltage is somewhere between 6 V and 20 V, the regulator will take care of the rest. You don't even have to worry about getting the polarity right – the power supply circuit doesn't care.

See page 25.

PS/2

A port for a PC keyboard and mouse.

See page 25.

LCD

A two-line LCD module with a maximum of 16 characters per line. An essential part of every proper prototyping board!

See page 24.

Audio I/O

With 16-bit stereo input and output, this prototyping board can also hold its own in the audio world.

See page 25.

Displays

The ideal way to display numbers. Also handy for displaying the date or time.

See page 24.

Pushbuttons

Besides all sorts of sophisticated inputs and outputs, simple operation using pushbuttons is often desirable.

See page 24.

DIP switches

For enabling or disabling options. Naturally, they can also be used as independent switches.

See page 25.

This prototyping board provides an environment for the FPGA module that enables it to do something useful. That gives you an opportunity to discover the capabilities of our FPGA module for yourself. Naturally, you don't have to make this voyage of discovery all on your own – a course in the form of a series of instalments in upcoming issues will guide you on your way.

We've designed a prototyping board that enables the FPGA to be used for a variety of tasks. As you can see from the list of features at the head of the article, it's equipped with quite a few modern interfaces.

For many of our readers, designing a circuit that will ultimately be implemented in an FPGA will be a totally new experience. For that reason, we'll start a series of articles on FPGA programming in next month's issue. It will be based on the prototyping board described here.

Just like the FPGA module, this circuit board is supplied in ready-to-use form.

>>>

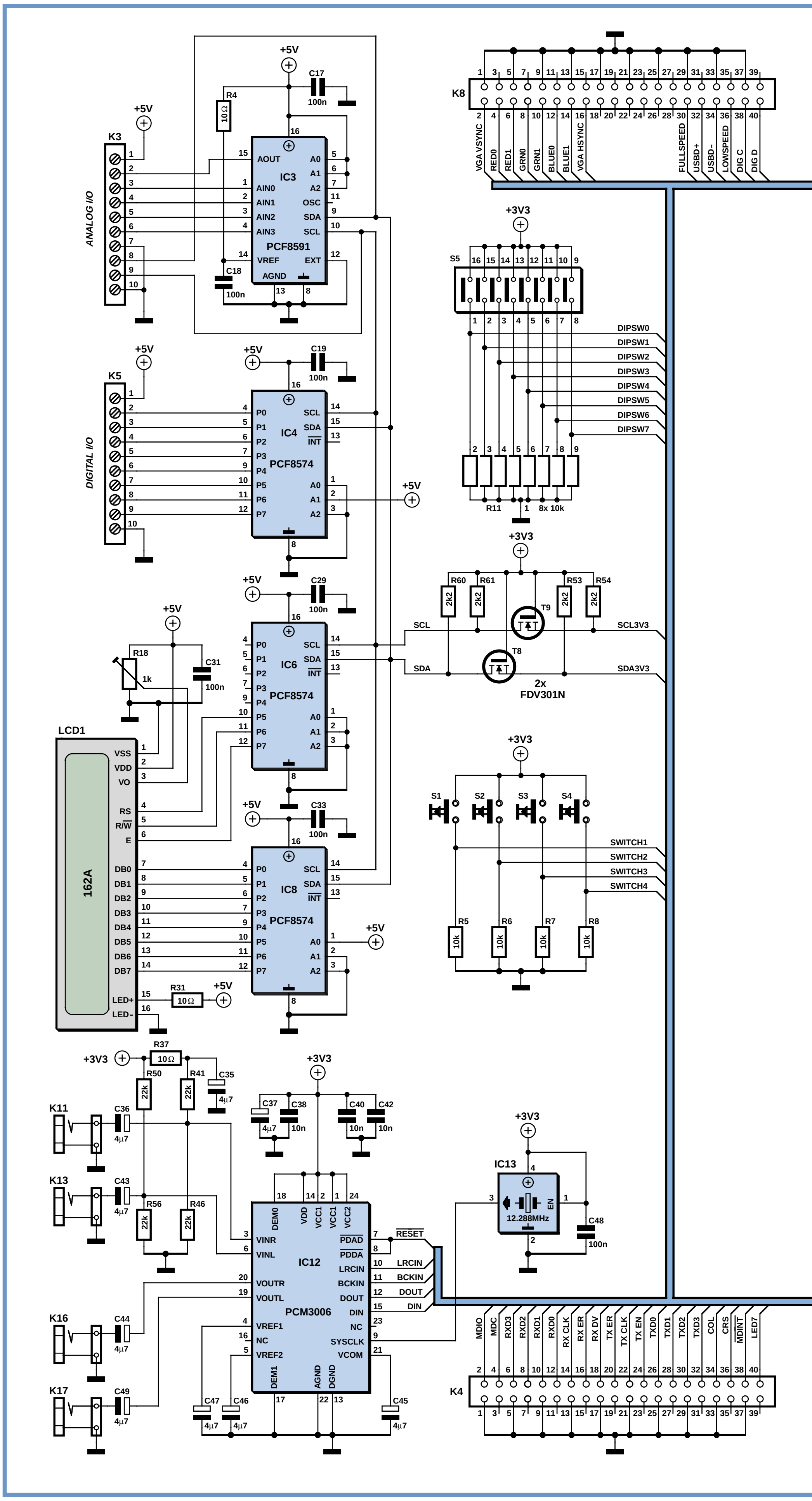
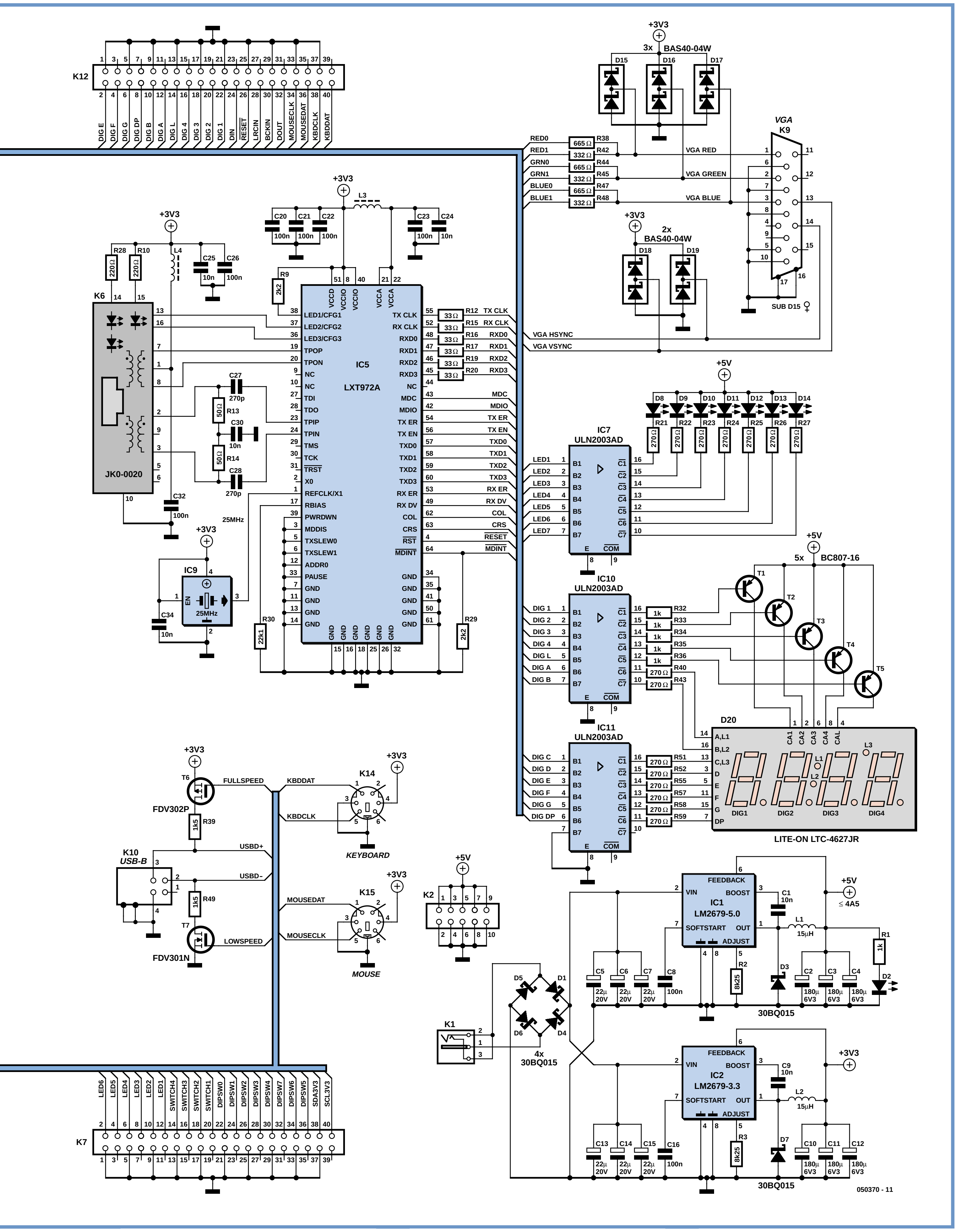


Figure 1. Schematic diagram of the FPGA prototyping board. The board layout is available free of charge in PDF format at www.elektor-electronics.co.uk.



VGA

Our prototyping board even has its own VGA output. The circuitry for this can be found around K9. Three super-cheap D/A converters are implemented using six resistors. The synchronisation signals can remain in digital form. As signal reflections can be expected here, all of these lines are protected against overshoots by a set of fast Schottky diodes (D15–D19).

Generating a VGA signal is easier than it appears. After reading the **seventh instalment** of the course, you'll know exactly how it's done.

Ethernet

The Ethernet interface is built around IC5, which converts digital signals into analogue signals and the other way around, all according to the Ethernet standard. Driving this interface is fairly complex. In the **ninth instalment,** we'll use this interface to perform some experiments on a network.

7-segment displays

The board has four 7-segment displays with decimal points, which are housed together with three separate LEDs in a single package (D20). This display has several common-anode leads (5 in total). They can be individually connected to the supply voltage to determine which of the five areas of the display should be illuminated. Here four of them are used for the four 7-segment displays and associated decimal points. The fifth lead provides current to the three separate LEDs.

The anode leads can be used to control which elements of these five areas light up.

For the drive signals, we use the same type of IC as for the LED drive signals. Several outputs of IC10 and IC11 are used to switch the anodes to ground under control of the FPGA. Five leads are used to switch transistors T1–T5. In this way, we can use signals DIG1–DIG4 and DIG L to control which part of the display is active.

Driving this display is somewhat difficult because the various parts of the display must be driven in rotation. We'll illustrate how to do this in the **second instalment** of the FPGA course.

Digital I/O

We can easily imagine that the prototyping board will be used as a control unit for something or other. There are eight digital I/O lines to allow user hardware to be connected to the prototyping board.

The IC we selected for this purpose (IC4) has an I^{2}C interface. This interface is also used to communicate with another IC that we'll describe shortly.

The circuit around T8 and T9 has already been used in another *Elektor Electronics* design. It converts the 3.3-V I^{2}C signals into 5-V I^{2}C signals.

The I^{2}C bus uses a serial data transmission protocol. The way this is implemented in the FPGA will be described in the **third instalment** of the course.

USB

A standard USB interface is built round connector K10. It can be optionally configured to present itself as a fast or slow USB device. The PC recognises a fast USB device by the fact that the D+ USB line is pulled up to the positive supply voltage by a resistor. For a slow device, the D– line of the USB is pulled to ground by a resistor.

On the prototyping board, either option can be selected by the FPGA as desired by switching FETs T6 and T7 on or off. Here again, the two data lines are wired directly to the FPGA.

We'll perform a few experiments with the USB bus in the **eighth instalment**.

Analogue I/O

Besides the digital I/O options, the board has four analogue inputs and one analogue output. These 8-bit I/O ports are provided by IC3, which also has an I^{2}C interface. For that reason, this I/O option will also be described in the **third instalment** of he FPGA course.

LEDs

LEDs D8–D14 can be driven directly from the FPGA module by signals LED1-LED7. IC7 buffers the drive signals. Whenever an input is at a logic High level, the corresponding output is pulled to ground. The currents through the LEDs are limited by resistors R12–R27.

This type of drive is very simple. In the **first instalment** of our FPGA course, we will use these LEDs to study various digital functions.

Pushbuttons

One of the input options of the prototyping board takes the form of pushbuttons. This is provided by pushbutton switches S1–S4. Signal lines SWITCH1–SWITCH4 are held low by resistors R5–R8 when the pushbuttons are not pressed. If a pushbutton is pressed, the associated signal line is pulled up to the supply voltage. Signal lines SWITCH1–SWITCH4 are connected directly to the FPGA module, just like the LED signal lines.

You can use the pushbuttons to control the states of four signal inputs to the FPGA. These four pushbuttons will also be used for simple input functions in the **first instalment** of the FPGA course.

DIP switches

Switch module S5 houses eight small switches, which are wired the same way as pushbuttons S1–S4. They allow an additional eight signal inputs of the FPGA to be controlled by the user.

A good application for switches of this sort is to use them for enabling or disabling various options. We will use these DIP switches in the **second instalment** of the FPGA course.

Audio I/O

The circuitry around IC12 forms a genuine 16-bit stereo input/output port. The sampling frequency is set to 48 kHz by the clock rate of IC13.

This IC communicates with the FPGA via five signal lines, which will be described in more detail in the **fourth instalment** of the course. Naturally, that will also include an example of an application in which the IC is driven by the FPGA.

LCD

The 2-line by 16-character LCD (LCD1 on the schematic diagram) is driven by IC6 and IC8. These ICs operate on the same principle as the ones used for the digital I/O lines.

We'll put the LCD through its paces in the **fifth instalment.**

PS/2

A PS/2 keyboard and mouse can be connected via connectors K14 and K15. Both signals on this bus are connected directly to the I/O pins of the FPGA module. As with most of the other I/O ports, the FPGA provides 'intelligent' control for these signals. Even the bidirectional data signal can be processed directly by the FPGA without the intervention of any external ICs. This communication uses a special serial transmission protocol, which we'll examine more closely in the **sixth instalment.**

Power supply

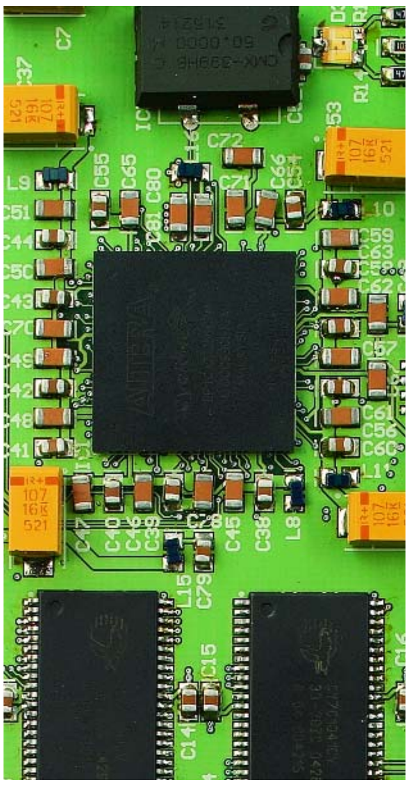
All of this is rounded out by the power supply. That's the only part of the circuit that isn't controlled by the FPGA.

IC1 and IC2 are both step-down controllers. Each of these ICs forms a step-down converter in combination with an inductor, a diode and buffer capacitors. The advantage of these converters is that a high input voltage doesn't cause the supply to dissipate a lot of power in the form of heat, unlike a series-regulated power supply using a 7805 or the like.

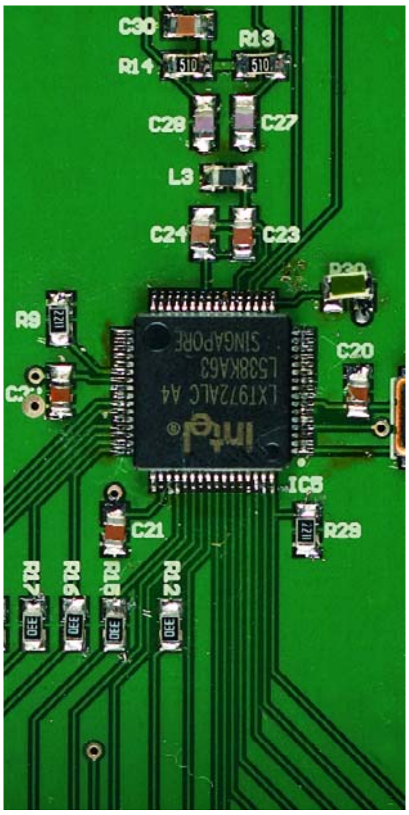
The input voltage on K1 can be connected either way round. D1 and D4–D6 sort things out so the polarity of the voltage at the inputs of IC1 and IC2 is always correct. The electrolytic capacitors at the input ensure that fast switching currents stay on the board where they belong. As a result, the power supply generates relatively little interference.

The FPGA is powered via K2.

(050370)

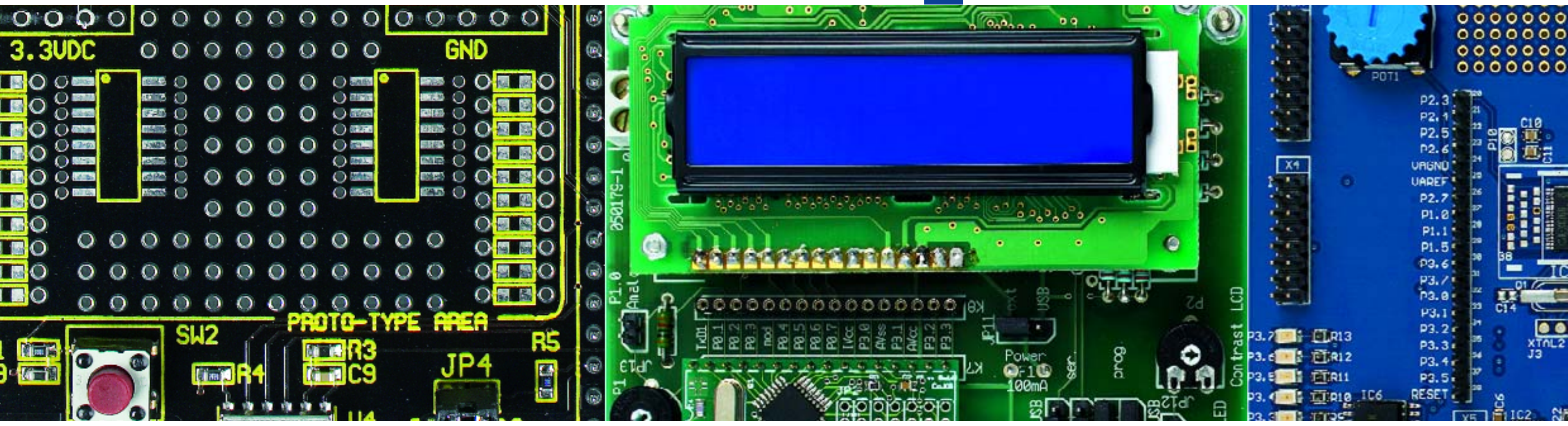


The "brains" on the Versatile FPGA Unit.



The Ethernet interface built around IC5.

Development



In this issue, which is bursting with FPGAs and microcontrollers, they could not possibly be omitted: development kits! We devote attention to interesting but affordable kits where we not only look at kits that are suitable for a commercial environment but which are also very interesting for hobby use.

The first few steps are always difficult. This is also true of course when working with FPGAs, microcontrollers and DSPs. Particularly in order to acquaint commercial developers in an easy way with new products, various chip manufacturers offer so-called ‘development kits’ (sometimes also called ‘starter kits’). But if you would like to use them at home for yourself, then that is generally not a problem at all. You can usually just order a kit from the appropriate distributor for private purposes. There is then nothing to stop you from messing about with your chosen kit. The only catch is that you ‘just’ have to decide which kit you would like to get started with. And this can be much more involved than you would think at first glance. In this article we attempt to give you an overview of interesting and affordable kits. We selected a few kits from each category and have summarised the important details in a table.

Options

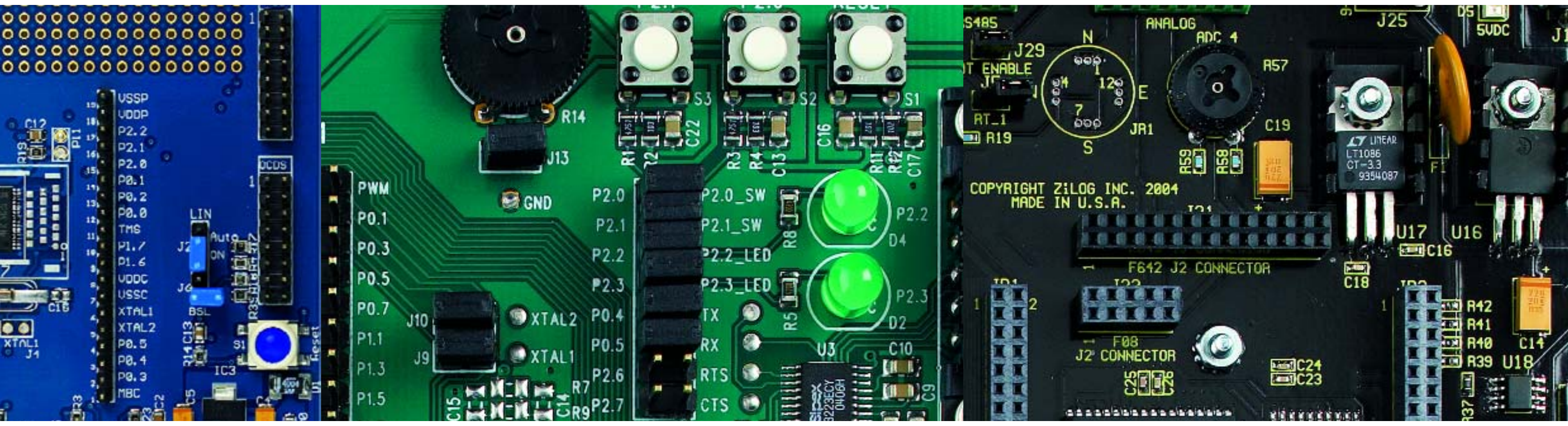
As already mentioned, before you start your development effort, you have to make a choice from the different options on offer from the various manufacturers. It is a good idea to not just consider what you need at this particular moment, but also to think ahead about possible extensions you may wish to add at a later date. Implementing a small design in an FPGA is generally not a problem, but a larger design (i.e., more gates) can be more trouble to make fit. That is why development kits are often equipped with one of the larger members, if not the largest member, from a family. For microcontrollers too, it is good to take into account your future requirements. Using fewer inputs and outputs is always possible. Wanting to use more than the IC provides is impossible.

Even the choice between an FPGA and a microcontroller may not be that immediately obvious. A microcontroller is less complex than an FPGA, but an FPGA, on the other hand, offers many more possibilities.
The development boards have a number of inputs and outputs for communication with other devices (that they may potentially control). The RS232 port remains a very familiar interface. In many cases headers are used, so that every pin from the IC can be accessed easily. Some boards have a number of ADCs and DACs and occasionally CAN, SPI and 1-Wire-interfaces make an appearance. For programming, each board has its own specific connector. Microchip uses a so-called ICD programming interface, while others use RS232, JTAG or some other specific interface. The same holds true here: try to think ahead of what you may need at a later stage.
In the large table we have included a number of critical characteristics so that it becomes relatively easy for you to make the final decision.
Nevertheless, we will provide some guidance to the starting developer by describing the broad characteristics of microcontrollers, DSPs and FPGAs.

Microcontrollers

A microcontroller is essentially a computer in one chip (refer **Figure 1**). The arithmetic unit is integrated together with all the I/O and memory in one IC, so that no additional chips are required (in contrast to a microprocessor which does need additional chips). Microcontrollers are used mainly for controlling electronic equipment, such as, for example, our recent SMD Oven (see *Elektor Electronics* January 2006).
The difference between an 8-bit and 16-bit controller is

Kits What's available, where do you start?



mainly the speed. A 16-bit controller can, compared to an 8-bit controller with the same number of MIPS (Million Instructions Per Second), process twice as much data. Most of the demo boards have been designed in such a way that they can function on their own. That means in most cases that there is a microprocessor on the board that runs a (demo) program. In addition there is then the possibility to simulate the board using the bundled software and, if the program is not quite 100 percent correct, use debug-mode to iron out the last wrinkles.
In the text box you can read some more regarding the choices for a particular type of microcontroller.

DSPs

DSP means Digital Signal Processor. Such a processor is really a kind of specialised microprocessor. The significant difference between a microcontroller and a DSP is that a DSP has been optimised for the mathematical operations that are necessary for digital signal processing. For example, a DSP has a special register structure and mechanisms to carry out an FFT (Fast Fourier Transform, a mathematical operation to analyse the frequencies in a signal) as quickly as possible and to move large blocks of data as efficiently as possible.
These days the boundary between DSPs and microcontrollers is becoming increasingly blurred, for example the PIC series from Microchip. The microcontrollers are then provided with the partial functionality of a DSP (which means additional instructions).

FPGAs

An FPGA (Field Programmable Gate Array) is, as the name implies, a 'field' with programmable logic gates and programmable internal connections (see **Figure 2**). It is a real millipede. An FPGA can be programmed so that it carries out all kinds of logical operations. Ranging

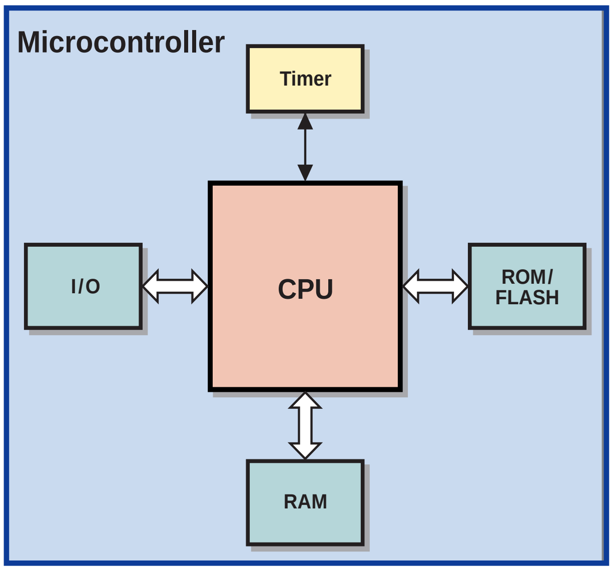


Figure 1. Block diagram for a typical microcontroller.

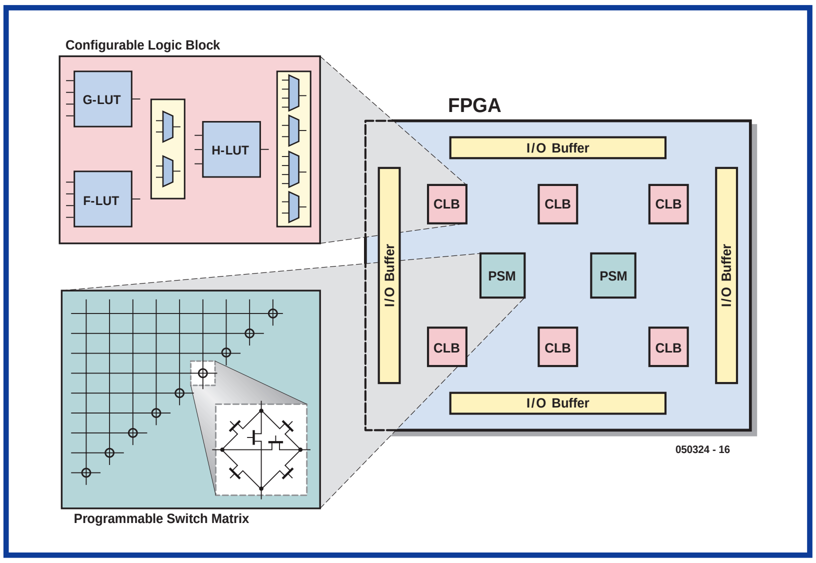


Figure 2. Typical architecture of an FPGA.

DEVELOPMENT KITS

Microcontrollers	Manufacturer / Website	Type of IC supplied	Programming Interface	I/O	LCD	Power Supply	R.R.P.
PICDEM 2 Plus	Microchip / www.microchip.com	PIC18F452 and PIC16F877	ICD	RS232	2x16	standard 9V DC adapter	£ 59
PICDEM 4	Microchip / www.microchip.com	PIC12F1320 and PIC16F627A	ICD	RS232; PIC16LF72 I/O expander	2x16	standard 9V DC adapter; 9V battery	£ 76
DS89C450-K00	Maxim/Dallas Semiconductor / www.maxim-ic.com	DS89C450	RS-232	2x RS232; 4x 8-bit I/O	-	6-9 V DC adapter	£ 62
Z8 Encore! XP 4K Series Development Kit	Zilog / www.zilog.com	Z8F0XXA (selectable)	USB or serial	RS232; IrDA, header for ADC input (up to 8 channels),	-	5V adapter supplied	£ 35
Z8 Encore! MCU Development Kit	Zilog / www.zilog.com	Z8FXX (selectable)	USB or serial	RS232, IrDA, header for ADC input (up to 8 channels)	-	5V adapter supplied	£ 35
SK-XC866 Starter Kit	Infineon / www.infineon.com	XC866	JTAG	RS232, CAN, LIN, JTAG; Headers for motor control and SBC board	-	8-18 V adapter	£ 100
eCOG1 Development Kit	Cyan Technology Ltd / www.cyantechnology.com	eCOG1	Parallell port	Fast Ethernet, parallel; debugger interface; 2x RS-232; I^{2}C, SPI, IrDA	2x16	adapter supplied	£ 200
AVR STK500	Atmel/ www.atmel.com	AT90S8515-8PC	RS232	2x RS-232; pin-headers for I/O ports of any AVR	-	10-15 V DC adapter	£ 90
R8C Starter Kit	Elektor/Glyn / www.elektor-electronics.co.uk; www.glyn.com	Renesas R8C-13 R5F21134FP#U0	RS232	USB; 2x serial port; LCD connnection	-	via USB or adapter	£ 62
DSP's							
dsPICDEM 2	Microchip / www.microchip.com	dsPIC30F4011	ICD	RS232; CAN; headers for all I/O ports	2x16	standard 9V DC adapter	£ 58
Explorer 16	Microchip / www.microchip.com	PIC24FJ128GA010 and dsPIC33F128GP710DSC	ICD, JTAG and PICkit 2	USB; RS232, JTAG; headers for all I/O ports; PICTail Plus	2x16	9-15 V DC adapter	£ 75
MAXQ2000-KIT	Maxim/Dallas semiconductor/ www.maxim-ic.com	MAXQ2000	JTAG	RS232, JTAG, 1-Wire interface; headers for all I/O ports	4 1/2	Via JTAG-interface, 5V and 6-9 V DC adapter	£ 55
FPGA's							
High Volume Starter Kit Bundle (comprising Spartan-3 Starter Kit and CPLD Design Kit)	Xilinx/www.xilinx.com	XC3S200-4FT256C (FPGA), XC9572XL-10VQ44C and XC2C256-7TQ144 Coolrunner-II (CPLDs)	JTAG	RS232; JTAG; PS/2 mouse / keyboard port, 3x 40-pin header	-	supplied	75
MAX II Development Kit	Altera / www.altera.com	MAX II EPM1270F256C5 (CPLD)	JTAG via ByteBlaster	USB; PCI; JTAG; Altera Expansion Prototype Headers	2x16	via USB- of PCI-bus	£ 105
ADDS-21261/Cyclone Evaluation Kit	Altera / www.altera.com	Cyclone EP1C3 (FPGA) and ADSP-21261 (SHARC DSP chip)	JTAG for FPGA, USB for SHARC	USB; RS232; JTAG; expansion header	-	via adapter	£ 140
EasyFPGA's EZ1KUSB Development Kit	EasyFPGA / www.easyfpga.com	Altera ACEX EP1K50TC144-3	USB or JTAG	USB; JTAG; 58 I/Os	-	via USB or supplied 6V DC adapter	£ 130
Morph-IC	Morph-IC / www.morph-ic.com	Altera ACEX EP1K10TC100-3	USB	USB; 2x20-pin header	-	via USB or supplied 5V DC adapter	£ 65

Devices Accepted	Manual	Software	Required for programming	Miscellaneous items supplied
18-, 28,- & 40-pin PIC16XXXX and PIC18XXXX	on CD	MPLAB IDE, MPASM, MPLAB C18	PRO MATE II, MPLAB PM3, PIC-START PLUS or MPLAB ICD 2	5k potentiometer, TC74 temperature sensor, piezo buzzer
8-, 14-, & 18-pin PIC16XXXX and PIC18XXXX	on CD	MPLAB IDE	PRO MATE II, MPLAB PM3, PIC-START PLUS of MPLAB ICD 2	NanoWatt technology/supercapacitor circuit, four 5k potentiometer, space for LIN transceiver and motor driver
DS89C430, DS89C440, DS89C450, DS5000	on CD	Microcontroller Tool Kit (MTK)	Software only; PC (MTK)	64kB Flash memory, 128kB SRAM
not exchangeable	on CD	ZDS II Integrated Development Environment, ANSI C-compiler	Supplied serial or USB Smart Cable	256-1k bytes RAM, 1-4k flash memory, 2 16-bit timers, comparator. Optional: 8 channel 10-bits ADC, temperature sensor
not exchangeable	on CD	ZDS II Integrated Development Environment, ANSI C-compiler	Supplied serial or USB Smart Cable	1-64k bytes Flash/ROM, 256-4k bytes RAM, up to 60 I/Os, up to 24 interrupts, up to 4 16-bit timers, up to 12 channels 10-bit ADCs, optional: DMA controller, SPI and I^{2}C.
not exchangeable	on CD	Evaluation versions of Keil uVision and Ulink or Hitex debugger Tantino-Eco, DAvE	Tantino USB (supplied)	Compatible with 8051, PWM generator, 10-bit ADC, 3 16-bit timers, 27 general purpose I/O, 768 bytes RAM, 16k flash, potentiometer
eCOG1	on CD	CyanIDE with ANSI C-compiler, simulator, debugger and Configuration Tool	Software only; PC	2MB 16-bit SDRAM, piezo buzzeer, 12-bit ADC, temperature sensor
8-, 20-, 28- and 40-pin AVR (Attiny, AT90S, ATmega)	on CD	AVR Studio	Software only; PC	2Mbit dataflash
not exchangeable	In Elektor / on website	KD30, NC30, HEW, Flash Development Toolkit	Software only; PC	8-bit timer, 12 channel 10-bit ADC, 5 external & 11 internal interrupts, 4kB flash, 10k potentiometer
18-, 28- and 40-pin dsPIC30FXXXX	on CD	MPLAB IDE	MPLAB ICD 2	potentiometer, temperature sensor, nine 10-bits ADC channels, SPI
PIC24 and dsPIC33 families	on CD	MPLAB IDE	MPLAB ICD 2	temperature sensor TC1047A, 10k potentiometer, 256kb EEPROM
not exchangeable		MAX-IDE	Software only; PC	MAX1407 ADC/DAC, potentiometer, JTAG interface board, LCD board
Spartan-3, Coolrunner-II, XC9500XL	On paper and CD-ROM	Evaluation versions of Xilinx ISE & EDK	Supplied JTAG3 cable	3-bit, 8-colour VGA display port, 1MB SRAM
not exchangeable	on CD	Quartus II Web Edition	ByteBlaster II parallel download cable	Temperature sensor, potentiometer 128kB SRAM, onboard power meterr
not exchangeable	on CD	Evaluation version Visual DSP++, Quartus II Web Version	Software only; PC	64 Mb SDRAM, 64Mb EE memory, 4Mb Flash
Altera ACEX EP1K10TC144-3, ACEX EP1K30TC144-3, ACEX EP1K50TC144-3	on CD	Quartus II Web Edition, USB-drivers	Software only; PC	
not exchangeable	on CD	USB drivers, FPGA loader program, Windows DLL (for use with Visual C++, Visual Basic, Borland Delphi), Quartus II Software Starter Suite	Software only; PC	Onboard 93C56 EEPROM

from basic logic gate functions (AND, OR, XOR, NOT, etc.) to complex combinational logic such as mathematical functions and decoders. It is even possible to emulate a complete (8051) microprocessor in an FPGA (provided that the FPGA has enough gates, of course).

These ports are formed by CLBs, which are connected in a matrix arrangement. Each CLB consists of multiple lookup tables (LUT), a few multiplexers and optionally a number of flip-flops. So the CLBs carry out all logic functions. By connecting the CLBs with programmable switches to each other in the right way, the desired functionality is obtained. The development software generally sorts out these interconnections for you, so you don't need to do much there yourself.

FPGAs have evolved from CPLDs (Complex Programmable Logic Device). As a consequence of their internal architecture, FPGAs have greater design flexibility compared to CPLDs. On the downside, the increased flexibility has also increased the complexity.

However, don't be taken aback by the overwhelming number of possibilities and accompanying data, descriptions, tutorials and other things. Once you've started with FPGAs, chances are that you cannot live without them any more.

(050324-1)

The right microcontroller for every user

Florian Schäffer

The first steps into the world of microprocessors seem to be easy at first glance. Countless starter kits tempt the prospective buyer with all sorts of bells and whistles, but on closer examination they tend to be much alike. A PCB with a microprocessor, eight or more I/Os, an RS232 port for communicating with a PC and an LCD for the displaying of text is pretty much standard. So when choosing the hardware you cannot really go wrong that much. But which one do you choose?

Apart from the differences in price, not too many things play a part when choosing a development kit for personal use. If you are just starting out in the world of microprocessors and are about to buy your first kit, it is a good idea to first think about what you would like to achieve with the development kit and how much scope there is for expansion. Are you only going to design a specific control system for which the number of inputs and outputs are already defined? Are you just having a tentative look around and want to try things out in a microprocessor environment, do a little bit of programming for some flashing LEDs and react to pushbutton inputs, and get an LCD to spring into life? Or do you need some special functionality, such as a serial port, a USB-interface or an I^{2}C-port for data exchange with other devices?

Depending on the purpose that you have in mind for the board, you will have to examine the various modules a little closer. It is best if you do not just rely on the brief technical datasheets from the manufacturer, but also look at the discussions in various Internet forums to read user experiences. In that way you get to find out the a particular board does have an RS-232 interface, but which is not very useful in real life because the 8-byte buffer is too small and the processor is much too slow to receive and process more than four symbols at data rates from 9600 baud and up.

With this we arrive at the next criterion: instruction set and speed of the processor. For a heating control application there is no need for the CPU to calculate at great speed. If it gets warmer a few seconds later, then that is not a big deal. It is different if a few thousand LEDs have to be driven via some multiplex scheme. If the CPU is not fast enough, the LEDs will just appear to flash in some meaningless fashion. If your program has to react quickly to signals 'from the outside', for example keyboard input or data from an interface, then it can be handy if the microprocessor is good at processing interrupts.

Also think about the various programming solutions. Do you need additional hardware to program the microcontroller, can it be (re-)programmed in the system (ISP – In System Programming)? What method do you prefer?

What are your ambitions regarding tinkering with these things? Most development kits consist of one PCB. The LCD is connected with a ribbon cable and the power supply consists of by a standard mains adapter, which is regulated on the board with a 780x. With very simple models it may even be possible that you have to provide your own regulated power supply.

Perhaps you're looking for something to do some control tasks in your home. A module that has been designed for rail/rack mounting, typically used in electrical installations, can be very practical. In order to realise your plans, it is probably necessary that you need more parts, which you will have to build yourself or that are possibly available read-made. Examples are a stepping motor driver or a relay board to control large loads.

After you have made your selection with respect to the hardware, do not forget to look at the software aspect. Of course, for most of the microprocessors some form of development environment is available. The question is, how much do you have to pay for it, and with which programming language do you have to work? Assembler is not something everyone is familiar with. C and BASIC are easier for beginners to use and are certainly not worse, although hardcore assembler hackers will sometimes have a good laugh about BASIC. The resulting machine code ultimately loaded into the processor is not substantially different compared to handwritten code. Because in the beginning you will probably need to rely on some help from other users, it is best to have a look for a suitable forum and see what sort of topics are being discussed. The best way is to use Google to search for forums that deal with a specific type of controller.

It is possible that a development kit contains a programming language that in practice is not actually used in combination with that kit, for example a C development environment. If you have questions in relation to that, it will take much longer to get an answer if other users use predominantly BASIC or assembler.

In the forums there are always people participating who are glad to help you out. Most of the beginner's questions have already been asked so it certainly pays to do a search of the forums first. In general you have to be prepared to do a little searching to obtain a satisfactory end result. You will also have to trawl through numerous datasheets.

The Triumphant March

30-year old design still inspires thousand

Roelf Sluman

Are eight-bit processors something from the past or is it still possible to do something useful with them? *Elektor Electronics* went looking and discovered that the good-old 6502, in a world of threaded computing and dual-core processors, still has a following of faithful fans.

In the 1970's and 80's three 8-bit processors dominated the market: the 6809 from Motorola, the Z80 from Zilog and the 6502 from MOS. By far the most popular of these three was the 6502: its low cost (when introduced, the 6502 set you back about 25 dollars) and the advanced design for its time made sure that the 6502 conquered the world in a short time as the brain in popular home-computers such as the Commodore 64 and the Apple II.

We are now some 30 years later and processors that are many thousands of times faster than the 6502 now dominate the market. But… that does not mean that there are no applications to be found for the elegant 8-bit processor. Many thousands of enthusiasts across the whole world still work daily with the 6502 and make it do things that were not considered possible in 1975.

Price war

When the 6502 was introduced in 1975 it cost about 25 dollars. That made it a serious competitor to the processor it was copied from, the 6800, which by comparison costs a whopping 179 dollars. No wonder computer manufacturers such as Apple and Commodore went for the 6502. Steve Wozniak from Apple had his eye in the 6800, but the enormous price difference ultimately forced the decision.

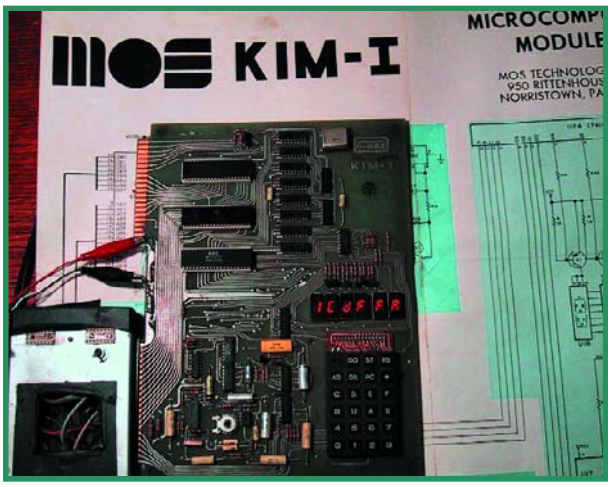


Figure 1. The KIM-1 computer, developed by MOS Technology. Input was via a hexadecimal keyboard, for the output a 6-digit LED display was present.

History

The 6502 processor celebrates its 30th anniversary this year. The introduction was preceded by a scandal: the designers of the 6502 had, in the first instance, developed another processor: the 6501. Unfortunately, it was a virtual copy of the 6800 processor from Motorola. This is not a surprise, because the same designers developed the Motorola 6800! Shortly after the 6800 processor appeared on the market, a dispute arose between Motorola and the 6800 designers. This conflict resulted in the resignation of most of the designers, who were promptly hired by MOS Technology (in the 70's Motorola's biggest competitor). MOS recognised the potential for the 6800 and asked the designers to design a processor that was pin-compatible with the 6800. That became the 6501, which was much cheaper because the development costs were negligible.

Naturally, Motorola weren't going to just let that happen and threatened to drag MOS into court. MOS responded with the 6502, which was exactly identical to the 6501 with the only change that the pin-out was different. As a consequence, the 6502 no longer fitted in boards that were designed for the 6800, which was reason enough for Motorola not to proceed with a lawsuit.

Because there were now no boards that would fit a 6502, MOS had to design something to bring the 6502 of the

s of designers

to the attention of software developers. This became the KIM-1 (see **Figure 1**), a single-board computer with 1 kB of RAM.

Computer manufacturers quickly appreciated the enormous potential that the 6502 offered. With the introduction of the Apple II and Atari 400 personal computers, the 6502 broke into the American market; Commodore was equally successful in Europe with the VIC-20 and later the Commodore 64 (with a more advanced version of the 6502, the 6510). The sales of just the Commodore 64 alone exceeds 25 million; the total number of 6502 processors sold worldwide is estimated at well over one hundred million!
In *Elektor Electronics* there were also countless projects using the 6502. The 'Junior Computer' was an exceptionally successful learning system published in the 1980s. From 1983 there was a series of boards that allowed you to build a complete 6502 computer. This finally resulted in 1985 in a system that was called Octopus 65.

The technology

The 6502 is a 5-volts, 8-bit processor with a 16-bit address bus, that with a range from 0x0000 to 0xFFFF is able to address up to 64 kB of memory. It comprises 4,300 transistors. The clock speed is 1 MHz, but because the 6502 does not have to work through a list of microcode instructions for each opcode that has to be executed, it is in practice just as fast as a 4 MHz Z80 (which does use microcode).

The instruction set consists of no more than 56 instructions and because RAM memory was much faster in those days, the designers saved on internal registers. The 6502 has therefore only three, each eight bits wide: the accumulator (the only register on which operations can be performed), and the index registers X and Y. The relatively large number of addressing modes of the 6502 makes it possible to address 64 kB of memory efficiently, using two-byte instructions (opcode and one operand). In order to fill 64 kB of memory, not more than 19 bytes of opcodes and operands are required. (Do you know how? Send the author an email: rsluman@gmail.com!)

There exist a number of variations of the 6502, with either more or fewer features. The best known are the 6507 (which was used in the Atari VCS2600 game computer) and the 6510 that can be found in the Commodore 64. The Commodore 64 was the first home computer with 64 kB of memory that could be switched in or out as desired in an intelligent manner, via a method called *bank switching*.. (For example, in the memory area into which the I/O ports were mapped there was also a section of RAM that could be switched into 'view' so that data could be copied to it and then switched 'out of view' so that the I/O-ports were accessible again.)

Get cracking yourself with a 6502

The best way to get to know the 6502 is to work with one yourself.

Simulator

The easiest way is using a *simulator*, a piece of software that imitates the behaviour of a 6502. An example of an excellent 6502 simulator is the program 6502 Simulator that you can download free from http://home.pacbell.net/michal_k/6502.html (Figure 2), among others.

Emulator

An emulator is a computer program that creates a so-called *virtual machine*: a computer inside a computer as it were. Because an emulator is a software replica of an authentic computer or processor, all software written for a certain computer or processor will also run in the emulator. Even possible design faults that were present in the

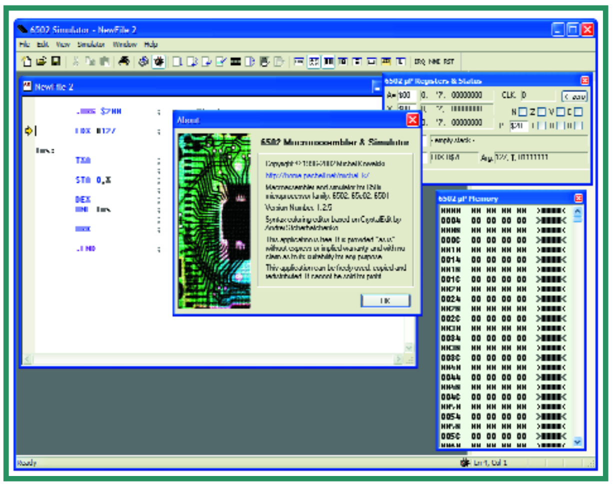
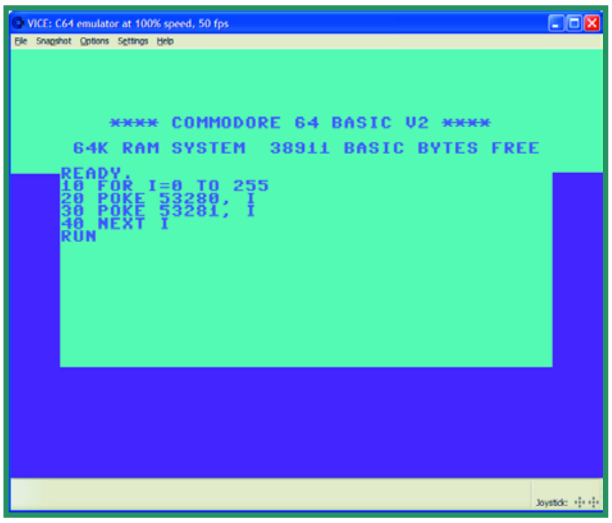

6502 Simulator - NewFile 2
File Edit View Simulator Window Help
IRQ NMI RST
NewFile 2
About
6502 Microassembler & Simulator
OK
6502 µP Registers & Status
6502 µP Memory
Ready
Ln 1, Col 1

Figure 2.
Ideal for those who want to work with a 6502: the 6502 macro-assembler and simulator from Michal Kowalski.

original hardware are present in the emulator! Practically every computer from the 70's and 80's has been emulated by now. In a large number of cases, the software producers from the then popular computer software have made their software freely available for use on these emulators. If you're ever thinking nostalgically about your time in front of your Apple, Atari or Commodore computer then you can relive those times with an emulator!

One of the better-known emulator programs is VICE (a rather silly abbreviation for Versatlle Commodore Emulator). This can be found at www.viceteam.org. VICE was written to emulate all known Commodore home computers (including the Commodore 64). The VICE emulator is so perfect that practically every piece of software ever written for the Commodore 64 runs flawlessly under VICE. After starting VICE, the famous blue screen of the Commodore 64 appears (**Figure 3**). From now on you're dealing with a real Commodore 64. Even the keyboard has been arranged differently and now corresponds to the Commodore 64 keyboard. The disk drive and cassette player, including any possible turboloaders are also completely emulated.


VICE: C64 emulator at 100% speed, 50 fps
File Snapshot Options Settings Help
Joystick:

```
    **** COMMODORE 64 BASIC V2 ****

 64K RAM SYSTEM  38911 BASIC BYTES FREE

READY.
10 FOR I=0 TO 255
20 POKE 53280, I
30 POKE 53281, I
40 NEXT I
RUN
```

Figure 3. The old, trusty screen from the Commodore 64, now as emulator on a modern computer.

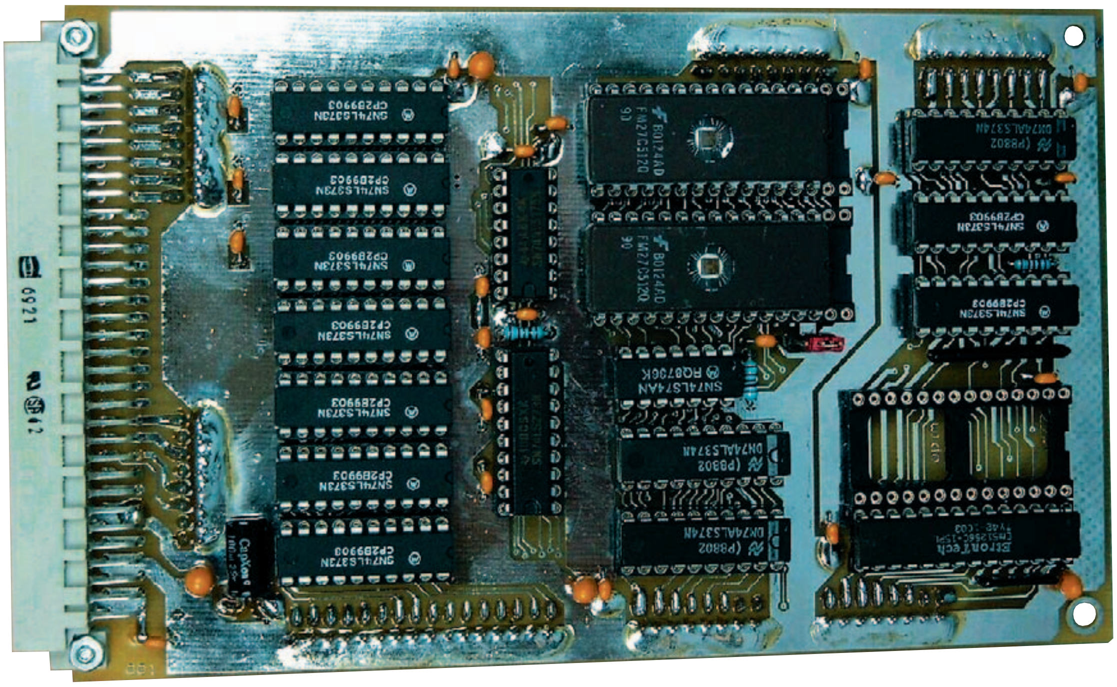


Figure 4. One of the two PCBs that comprise the 6502 built from discrete parts by Dieter Müller. (source: website Dieter's Hobby Projects)

Programmable Gaming Hardware

The Amiga and Atari ST under the logic analyser

The page is to log my progress implementing Amiga and Atari computers including the 68000 cpu core.

VHDL Source code available for download one day .....

A complete Commodore Vic-20 living in an FPGA

The classic Vic 20 home computer from the 1980's recreated in a modern FPGA.

VHDL Source code available for download

Bally Astrocade

The Bally Professional Astrocade first hit the streets of the USA in 1979 and survived until around 1985. The console uses the same chips as the arcade games Gorf and Wizard of Wor.

VHDL Source code available for download

Figure 5. At FPGA Arcade various 6502 hardware for games are emulated in an FPGA.

Copying

You can also copy a 6502 using 'discrete' components such as 7400-series ICs, RAMs and EPROMs. This is what Dieter Müller did with some old bits he had at home. He nearly succeeded, using about 40 ICs and two PCBs (Figure 4). The most important concession is a different timing, but his processor knows all 6502- and 65C02 opcodes (see http://people.freenet.de/dieter.02/m02.htm).

A different approach is to start with an FPGA and to build into it the complete functionality of a 6502. An

FPGA, after all, contains a large number of programmable logic blocks and in a modern version the circuit for an 6502 can be easily accommodated (or even multiple copies in one FPGA!). There is an organisation, which goes by the name Opencores, that is engaged in the development of various CPU cores in FPGAs (http://opencores.nnytech.net). The 6502 code can be found after a bit of searching, it is called T65. This is also available from http://www.fpgaarcade.com/, where a number of games variations of the 6502 are available as well (**Figure 5**).

Applications

Because the 6502 is such a versatile and cheap processor, and because it is easy to combine with other hardware, it is still a favourite element in many DIY projects. Many projects can be found on the Internet that use a 6502. Most of these comprise a home-built computer with the 6502 as its nerve centre, but a number of very inspiring applications can also be found...

The 6502 in the casino

That a 6502 can be used to make a lot of money was discovered by the 'Eudaemons', a pair of American students from California. They built two circuits based in the 6502, which they housed in two shoes. They subsequently went to a casino and picked a roulette table. Student number one moved his shoe in the same rhythm as the revolutions of the roulette wheel. The 6502 processor in this shoe used a revolutionary algorithm to calculate on which number the ball would come to rest. This information was then automatically sent to student number two, who could determine, based on the vibrations in his shoe, on which number to bet.

Give your robot eyes

Mike Naberezny used a combination of four Sharp GP2D02 infrared sensors and a 6502 to give his robots 'eyes'. The program does nothing more than continually poll the four sensors (high values mean 'close' and low values mean 'far') and pass this information to the robot. On the Internet (with as starting point www.6502.org) many more such fantastic projects around the 6502 can be found.

(050316-1)

2 kB memory? More than enough!

The first computer in which the 6502 was used was the world famous games computer from Atari, the VCS2600. Although the 6502 was already the cheapest computer in its class, Atari managed to reduce the price even further by ordering a special version of the 6502, the 6507. This processor could address a maximum of 8 kB instead of the normal 64 kB. Not a problem, according to the designers of the Atari 2600. After all, memory was so expensive that no computer game would ever use more than 2 kB.

Where can you find a 6502 now?

Gunther Ewald

It is hard to believe, but microcontrollers with a 6502 core are still being produced today.

Manufacturers such as Micronas and Renesas use this CPU core in cheap microcontrollers for simple control tasks.

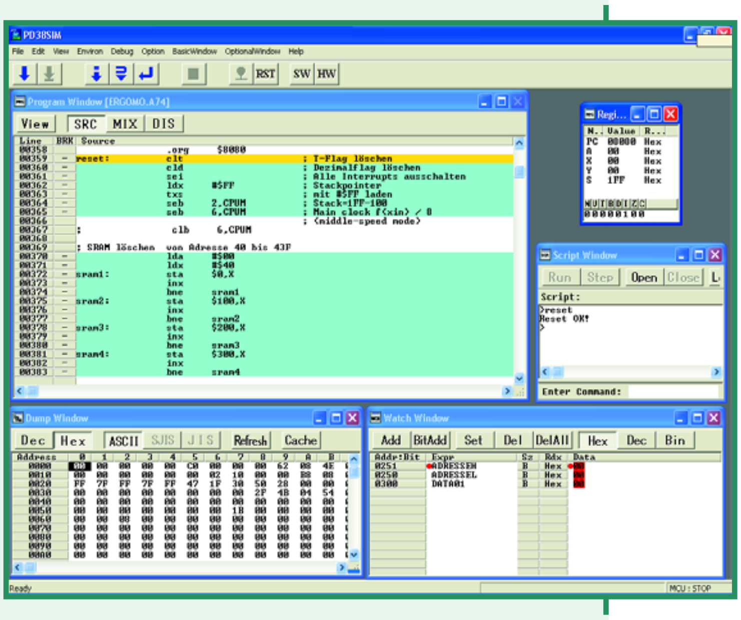
Have a look at Micronas at the WDC65C816 CPUs, which are used mostly in automotive applications:

http://www.micronas.com/products/by_function/cdc_1607f-e/product_information/index.html

Renesas employs the 6502 core in the entire 740 family. These are (with exception of specific extensions) completely opcode-compatible with the original.

Unfortunately the selection has been reduced somewhat as a result of switching to lead-free production, but the manufacturer still has 37 derivatives with a 6502-core:

www.renesas.com/fmwk.jsp?cnt=740_family_landing.jsp&fp=/products/mpumcu/740_family/


PD38SIM
File Edit View Environ Debug Option BasicWindow OptionalWindow Help
RST SW HW
Program Window [ERGOMO.A74]
View SRC MIX DIS
Regi...
Script Window
Run Step Open Close
Dump Window
Dec Hex ASCII SJIS JIS Refresh Cache
Watch Window
Add BitAdd Set Del DelAll Hex Dec Bin
Ready
MCU : STOP

```
Line   BRK  Source
                        .org    $8080
        -   reset:      clt                     ; T-Flag löschen
        -               cld                     ; Dezimalflag löschen
        -               sei                     ; Alle Interrupts ausschalten
        -               ldx     #$FF            ; Stackpointer
        -               txs                     ; mit #$FF laden
        -               seb     2,CPUM          ; Stack=1FF-100
        -               seb     6,CPUM          ; Main clock f(xin) / 8
                                                ; (middle-speed mode)
            ;            clb     6,CPUM

            ; SRAM löschen  von Adresse 40 bis 43F
        -               lda     #$00
        -               ldx     #$40
        -   sram1:      sta     $0,X
        -               inx
        -               bne     sram1
        -   sram2:      sta     $100,X
        -               inx
        -               bne     sram2
        -   sram3:      sta     $200,X
        -               inx
        -               bne     sram3
        -   sram4:      sta     $300,X
        -               inx
        -               bne     sram4
```

```
N..  Value  R...
PC   08080  Hex
A    00     Hex
X    00     Hex
Y    00     Hex
S    1FF    Hex
N U T B D I Z C
0 0 0 0 0 1 0 0
```

```
Script:
>reset
Reset OK!
>

Enter Command:
```

```
Address  0  1  2  3  4  5  6  7  8  9  A  B
 0000   00 00 00 00 00 C0 00 00 00 62 00 4E
 0010   00 00 00 00 00 00 02 10 00 00 B8 00
 0020   FF 7F FF 7F FF 47 1F 30 50 28 00 00
 0030   00 00 00 00 00 00 00 00 2F 4B 04 54
 0040   00 00 00 00 00 00 00 00 00 00 00 00
 0050   00 00 00 00 00 00 00 1B 00 00 00 00
 0060   00 00 00 00 00 00 00 00 00 00 00 00
 0070   00 00 00 00 00 00 00 00 00 00 00 00
 0080   00 00 00 00 00 00 00 00 00 00 00 00
 0090   00 00 00 00 00 00 00 00 00 00 00 00
 00A0   00 00 00 00 00 00 00 00 00 00 00 00
```

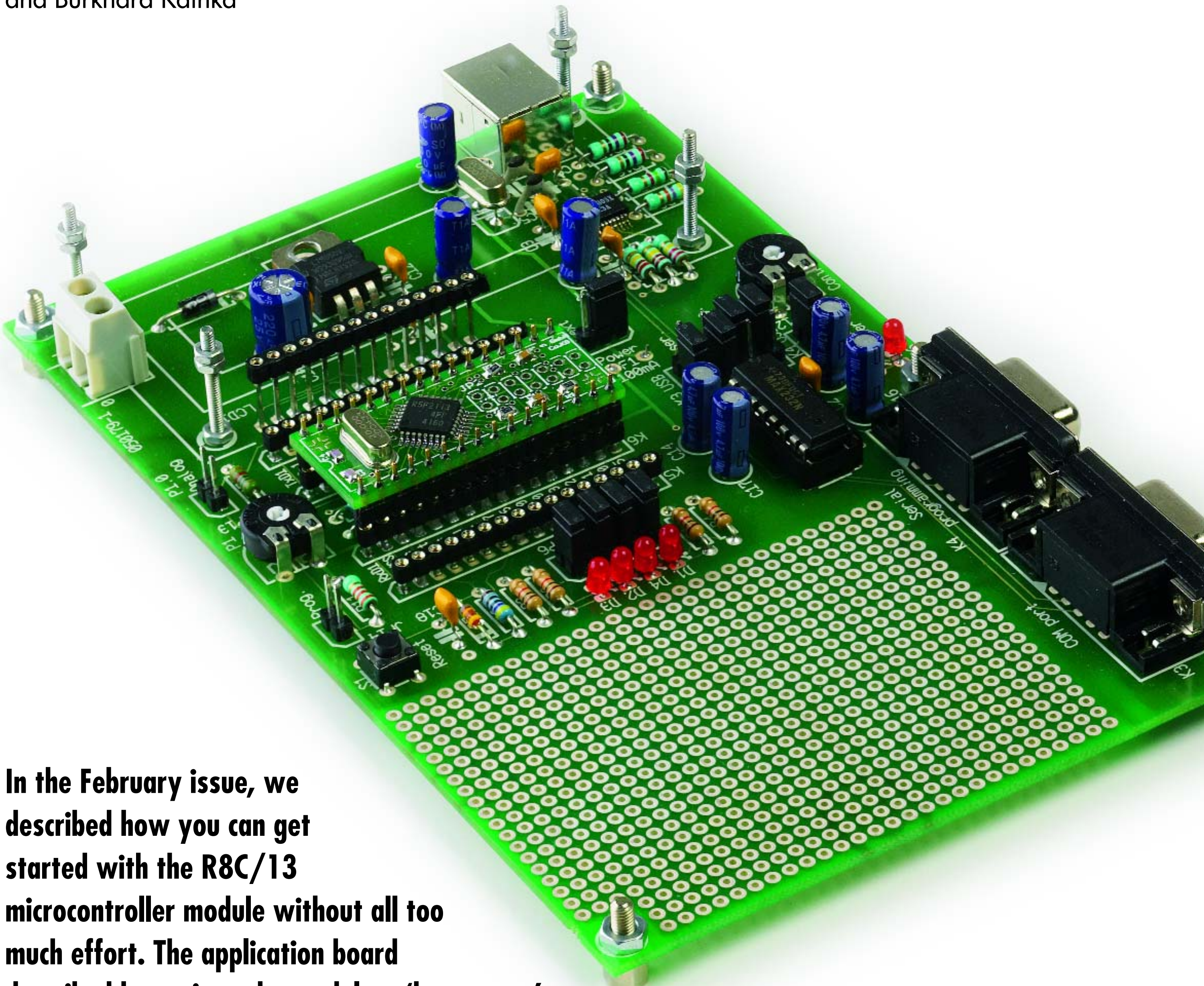
```
Addr:Bit  Expr        Sz  Rdx  Data
0251      ADRESSEH    B   Hex  00
0250      ADRESSEL    B   Hex  00
0300      DATA01      B   Hex  00
```

These products are not really suitable for hobby applications. You need to use a special emulation MCU and an emulator. The manufacturer does make a free MCS-simulator available with a runtime of 4 months. This offers I/O-simulation, interrupt-simulation, execution cycle measurement, RAM monitor display and coverage measurement. The program is available from the Renesas homepage.

Application board

Base camp for high endeavours

Gunther Ewald
and Burkhard Kainka



In the February issue, we described how you can get started with the R8C/13 microcontroller module without all too much effort. The application board described here gives the module a 'base camp' with lots of connections to the outside world, including a USB interface and two serial ports. There's also an LCD interface and other useful features for developing your own applications.

for R8C/13

The starting point for this design was the question, what do you need to successfully utilise all the major features of the microcontroller with a minimum of effort? The result is a circuit board with the following complement of functions:

- two serial interface ports
- a USB port using an integrated USB/serial adapter
- an LCD interface
- four LEDs that can be connected to port lines
- a potentiometer connected to one of the many analogue inputs
- a power supply socket and 5-V voltage regulator
- an option for powering the board via the USB port
- a reset switch
- a jumper for the MODE input

The USB interface on the board deserves special mention. Serial interfaces are becoming increasingly rare on new PCs, but asynchronous serial interfaces are fundamentally necessary for communi-

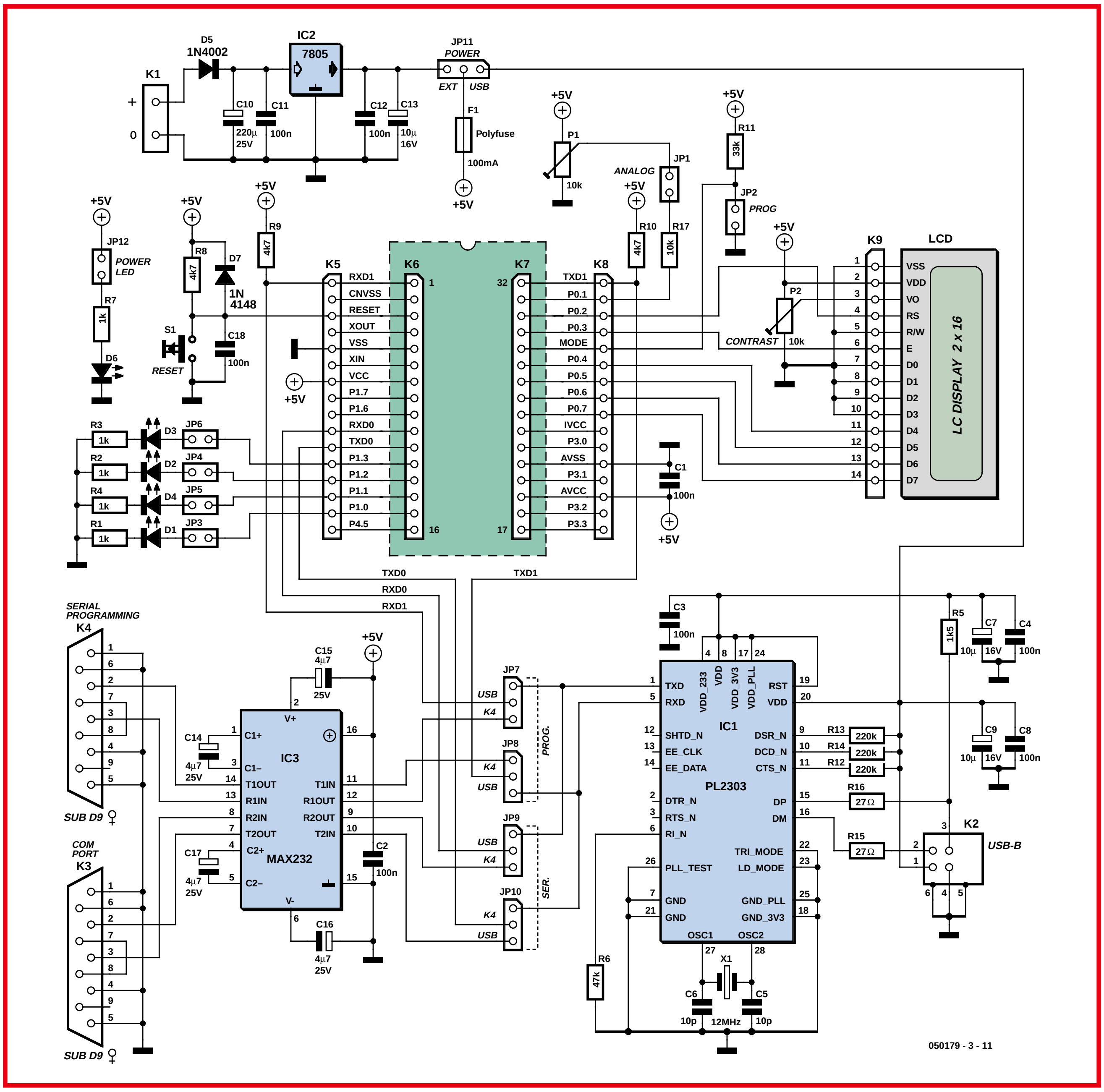


Figure 1. The circuitry on the application board provides two serial ports, a USB interface and an LC display to connect the R8C daughter board to the outside world.

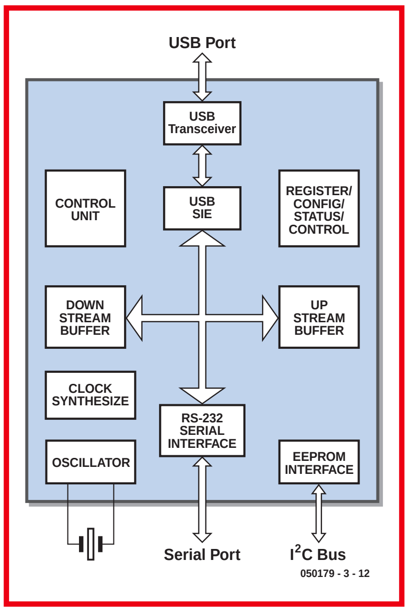


Figure 2. **Block diagram of the Prolific interface converter.**

cation with the microcontroller. For that reason, we decided to simply include a USB/serial adapter on the board.

In contrast to the equally viable approach of using an external USB adapter, this has the advantage of allowing the board to be powered via the USB port. A total of four jumpers determine which serial interfaces of the microcontroller are routed via the RS232 ports or the USB port. For instance, you can load and debug programs via the USB interface, or you can use the USB interface for serial communication with the microcontroller and your programs.

Microcontroller ports

The schematic diagram (Figure 1) essentially shows the principal features of an R8C system as already mentioned in the February issue. The MODE jumper and Reset switch are important features. Interface drivers are also necessary for the RXD1 and TXD1 leads of the debug interface, which are routed here via JP7 and JP8 to allow either RS232 or USB to be used. Depending on the jumper settings, you can use either the MAX232 and connector K4 or the PL2303 USB controller and a USB link.

In addition to the serial debug interface (UART1), the R8C/13 has a second, fully independent serial interface (UART0) that communicates via the TXD0 and RXD0 leads. As is well known, the MAX232 can handle more than just two lines – in fact, it can handle four. That's why jumpers JP9 and JP10 and a second serial interface connector (K3) are on the board. You can also select either RS232 or USB for the UART0 interface.

Now for a few words about the microcontroller ports. The microcontroller leads are routed to single-row socket headers in order to leave all options open. However, specific functions have also been assigned to most of the port leads. Here 8-bit port P0 is intended to be used for connecting the LCD module, but it can of course be used for some other purposes if you don't need an LCD. A standard alphanumeric LCD can be driven in 4-bit mode using only six port pins. Leads P0_2 to P0_7 are thus used by the LCD.

P0_0 is reserved for the debug interface (TXD1) and is only available for user applications if the debug function is not used. That still leaves P0_1. As this lead can also be used as an analogue input, a potentiometer providing a variable input voltage of 0–5 V can be connected to it via JP13. A supplementary protective resistor (R17) prevents overloading if P0_1 is accidentally configured as an output while connected to the potentiometer. The two jumper pins can also be used as a measurement input. If you want to connect a sensor that outputs both positive and negative voltages, you can use the potentiometer to adjust the zero point to where you need it.

The four lower bits of Port 1 can be connected to the four LEDs via jumpers and series resistors. That's ideal for your initial programming exercises, such as those described in the February issue. If you haven't already tried the 'blinker' example program, now's your chance to get up to speed.

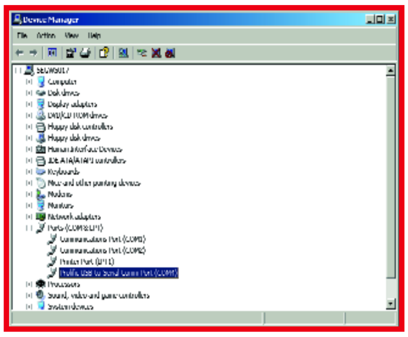


Figure 3. **The virtual COM interface in the Windows XP Device Manager window.**

If you remove jumpers JP3–JP6, these leads can also be used as four additional analogue inputs. Just add a few lines of code, and you have a four-channel digital multimeter with LCD display!

A total of seven individual port leads have been left open, but you're bound to think of a use for them. Applications such as an I_C or SPI interface and the like can always use an extra lead or two.

It takes a bit of juice

There are two options for powering the board, which are selected using JP11. Either you can use the power supply connector (K1) and voltage regulator IC21, or you can use USB with its built-in 5 V supply voltage.

If you're using USB anyhow, it makes sense to dispense with an external power supply. However, a bit of caution is advisable here. According to the USB specification, every USB down port is supposed to be protected against overloads by a PolySwitch protective device. However, painful experience proves that it just ain't so. A simple short on the USB supply line results in a small puff of smoke from the PC motherboard and causes the corresponding down port to be permanently disconnected from the supply voltage. For cost reasons, manufacturers obviously do nothing more than placing small series resistors in the supply tracks, which then simply melt through.

For this reason, a PolySwitch protective device (F1) rated at 100 mA is included on the application board for the specific purpose of protecting the PC. That creates peace of mind and allows you to experiment in a relaxed mood.

USB interface

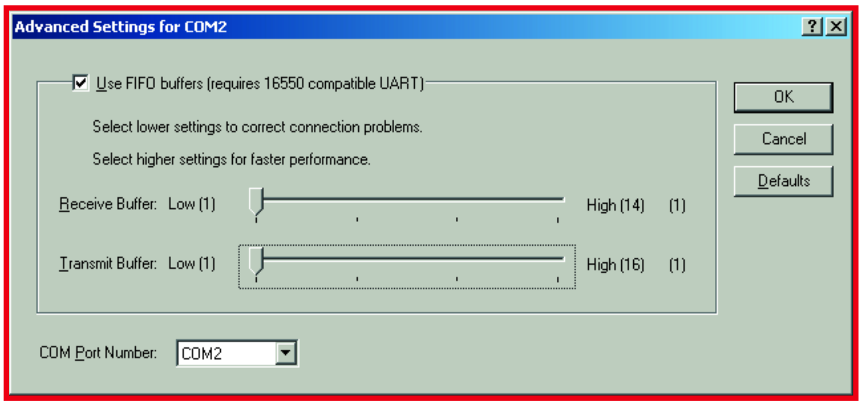
Next we come to the USB interface. The PL2303X, which sports the designation 'USB to Serial RS232 Bridge Controller', comes from the Taiwanese manufacturer Prolific (sales rep: Glyn). Attentive readers may recall having already seen this Prolific IC in a built-in, encapsulated form in an interface converter cable described in the September 2005 issue of *Elektor Electronics*.

This IC is a full-speed USB device for USB 1.1 and thus compatible with the more recent USB 2.0 interface. That means the data transmission rate on the bus is 12 Mbit/s. R5 connects the

D+ lead of the USB connector to 3.3 V to signal a full-speed device to the PC. That causes the PC to load a driver for a virtual serial interface. Such a driver allows serial interfaces to be operated at transmission rates up to 1,228,800 baud (that's 1.2 Mbaud!). In line with the capabilities of the R8C/13, various standard baud rates are used here. The virtual serial interface behaves exactly the same as a real RS232 interface for downloading and debugging programs. **Figure 2** shows the internal structure of the IC, which consists of a USB transceiver, a serial interface, and two data buffers. The data buffers support fast, uninterrupted data transfers, with data blocks being packed into individual USB frames. It is also possible to connect an I_C EEPROM to store user configuration settings, but that option isn't used here.
The non-inverted RXD and TXD signals are available at the serial side of the converter and can be connected directly to the R8C. The inputs of the PL2302X are compatible with 3.3-V and 5-V logic levels. The TXD, DTR and RTS outputs provide either 3.3-V or 5-V logic levels, depending on the voltage connected to pin 4. In this case, this voltage is 3.3 V from the internal voltage regulator of the PL2303. That means the level applied to the serial input of the R8C/13 is 3.3 V, although the level actually expected there is 5 V. That's permissible, though, because the R8C/13 can also be operated at 3.3 V, and besides that a 3.3-V signal at the RXD input of the microcontroller is adequate even when it's operated at 5 V. All inputs of the R8C regard any voltage above 0.2 V_{CC} as a logic 'high' level.

Driver installation

You may have already used a Prolific USB adapter with your system. In that case, the interface will be recognised as soon as the board is connected to the PC. If it isn't, or if you experience problems using the USB port, you should install the latest version of the driver. You can obtain the driver by downloading the file **050179-3-11.zip** from the Elektor Electronics website (www.elektor-electronics.co.uk). It contains the file 'PL-2303 Driver Installer.exe'.
Run the driver installer file before connecting the board. Any older version of the driver present in the PC will be deleted automatically when the new driver is installed. The next time the board is connected (or another interface converter using the Prolific IC is connected), Windows will automatically locate the appropriate driver and load it. After that, your PC will have an additional serial interface that can be used just like a normal RS232 interface.
If you have already installed several such interface converters, Windows will assign a high COM number to the new virtual interface. In that case, it's recommended to use Advanced Settings to rename the interface, for example to 'COM2'. You can usually ignore any message saying that this interface is already in use, because it refers to another virtual interface that is not used at the same time. All you need to know is which serial interfaces are actually present in your PC in hardware form. For example, the PC used for our tests had only COM1 present on the motherboard, but it also had an interface card with COM3 and COM4. In that case it's a good idea to name the virtual interface 'COM2' (see **Figure 3**), especially since many older programs only support COM1 and COM2. Problems with high COM numbers have also been observed with the FDT download program, so you should at least keep the number in the single-digit range.
In some cases it may be necessary to configure the FIFO buffer of the virtual interface to the smallest possible size (**Figure 4**) in order to obtain reliable communication. That's because with a large buffer size, it's theoretically pos-


Advanced Settings for COM2

☑ Use FIFO buffers (requires 16550 compatible UART)

Select lower settings to correct connection problems.

Select higher settings for faster performance.

Receive Buffer: Low (1) High (14) (1)

Transmit Buffer: Low (1) High (16) (1)

COM Port Number: COM2

Figure 4. Configuring the buffer sizes and COM port number.

COMPONENTS LIST

Resistors:
R1-R4,R7 = 1kΩ
R5 = 1kΩ5
R6 = 47kΩ
R8,R9,R10 = 4kΩ7
R11 = 33kΩ
R12,R13,R14 = 220kΩ
R15,R16 = 27Ω
R17 = 10kΩ
P1,P2 = 10kΩ

Capacitors:
C1-C4,C8,C11,C12,C18 = 100nF
C5,C6 = 10pF
C7,C9,C13 = 10µF 16V radial
C10 = 220µF 25V
C14-C17 = 4µF7 25V radial

Semiconductors:
D1-D4,D6 = LED
D5 = 1N4002
D7 = 1N4148
IC1 = PL2303X (Prolific)
IC2 = 7805
IC3 = MAX232

Miscellaneous:
JP3-JP6,JP12 = jumper
JP7-JP11 = 3-way SIL pinheader
K1 = 2-way PCB terminal block, lead pitch 5mm
K2 = USB socket, type B for PCB mounting
K3,K4 = 9-way sub-D socket, angled pins, PCB mount
K5-K8 = 16-way SIL pinheader
K9 = LCD module, 2x16 characters
S1 = pushbutton with 1 make contact (e.g. TS695)
F1 = 100 mA Polyfuse
X2 = 12MHz quartz crystal
PCB, bare, order code 050179-1
PCB, ready-assembled and tested, order code **050179-92**
LCD with backlight, order code 030451-72
Poly-LED display, order code 030451-73

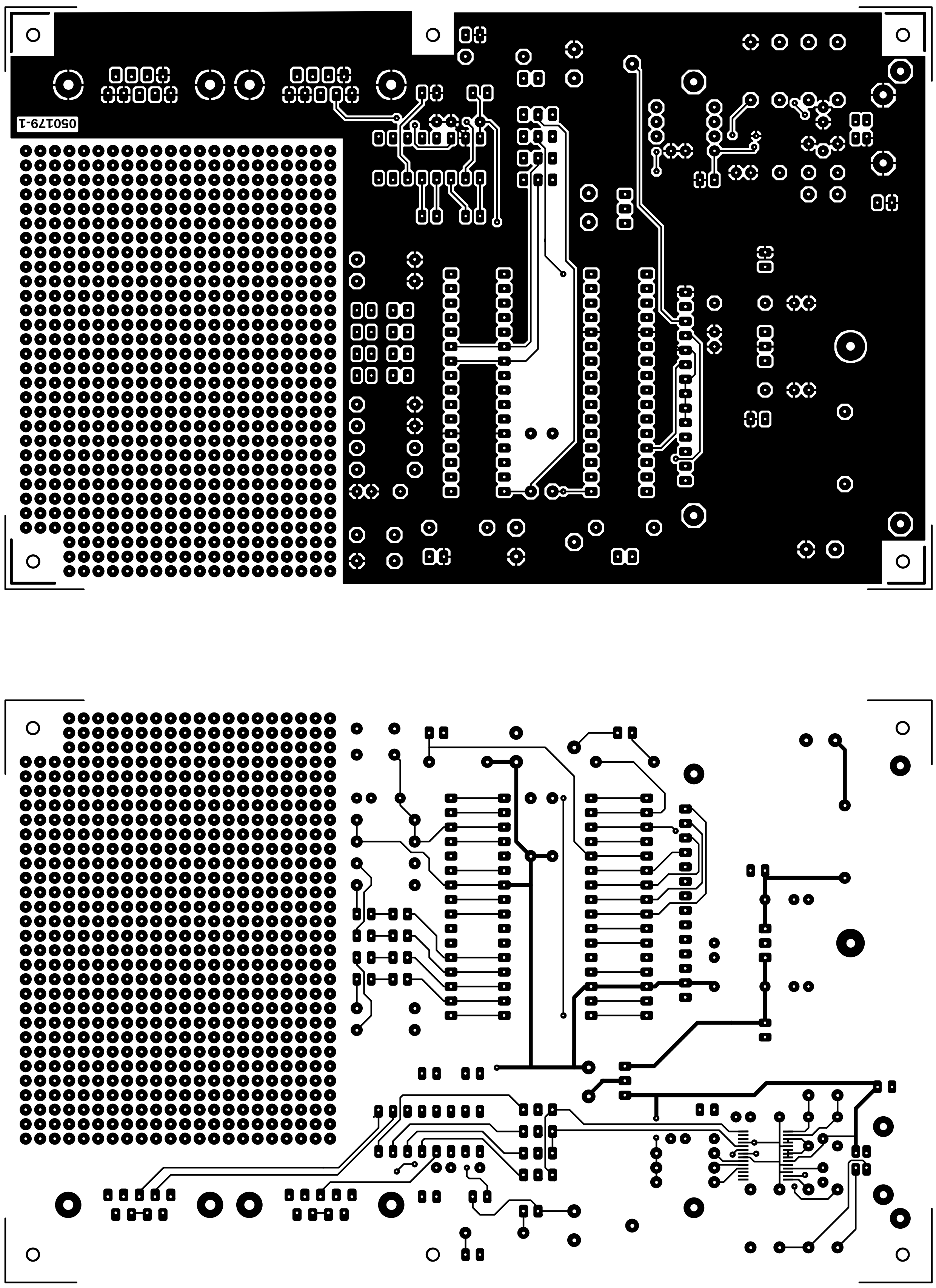


Figure 5a. The track and ...

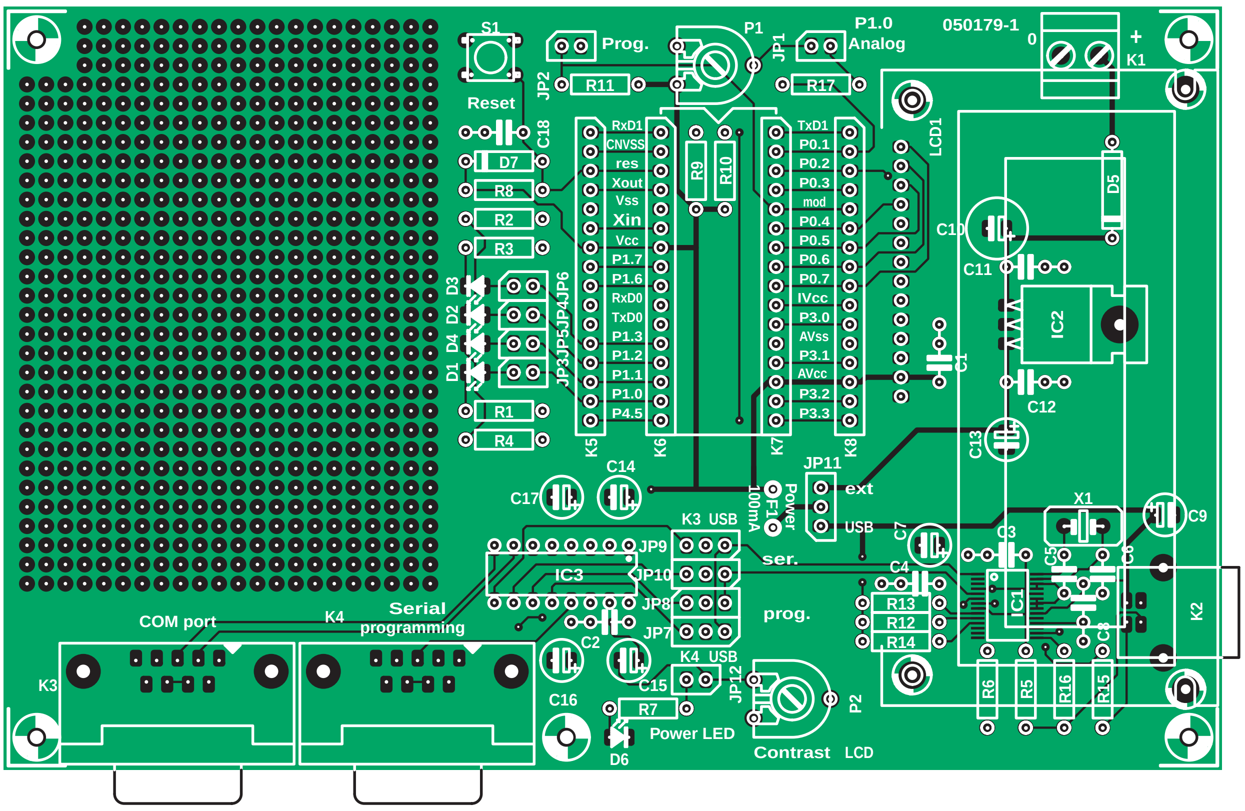


Figure 5b. ... component layouts of the double-sided circuit board.

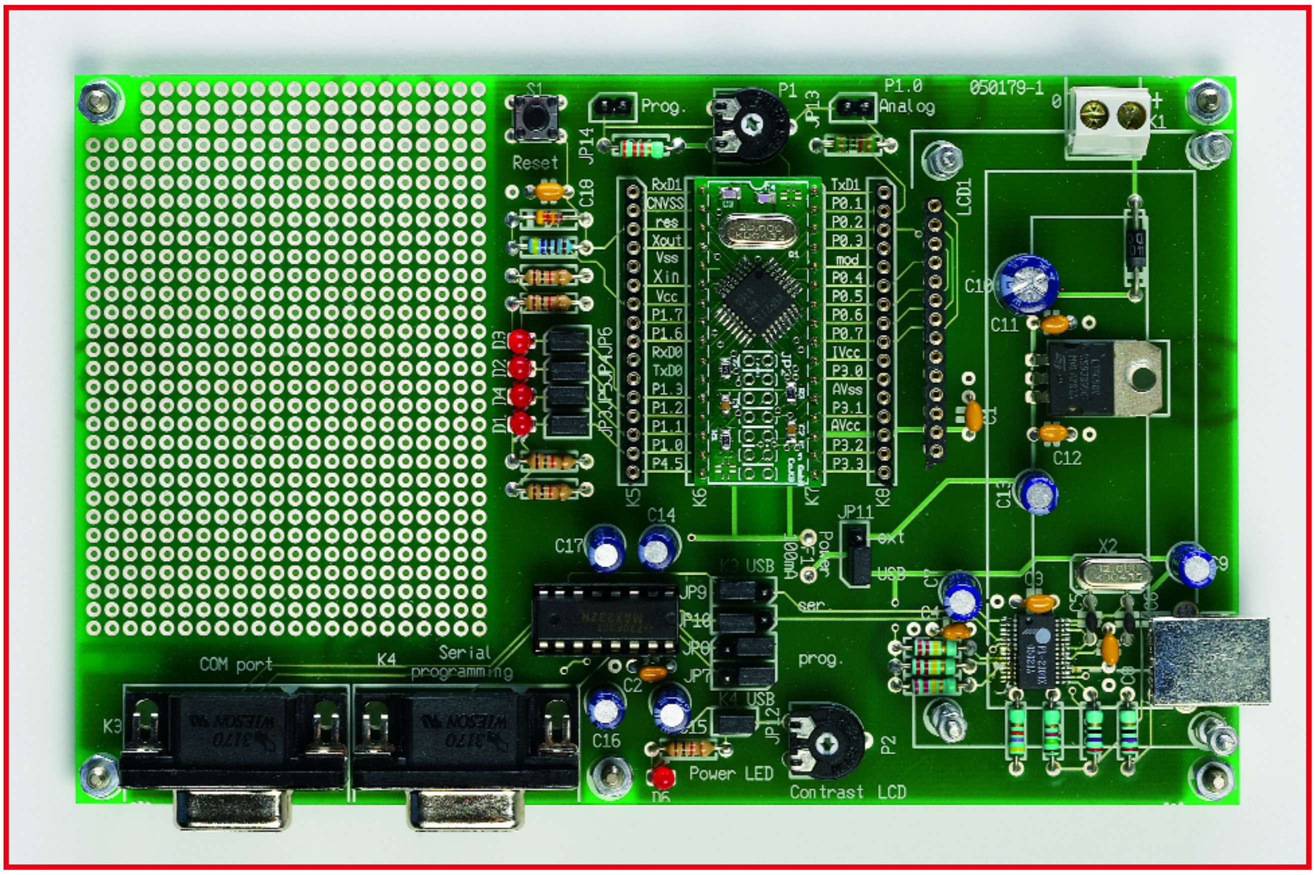
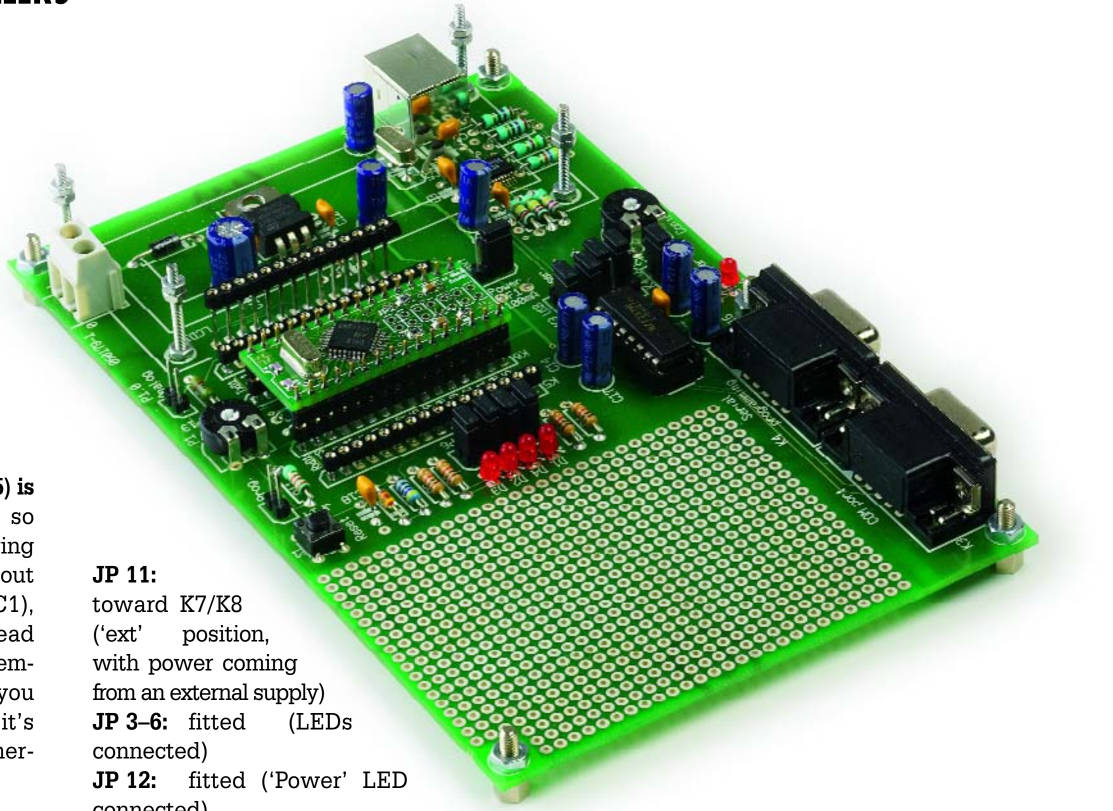


Figure 6. The fully assembled prototype board. The component overlay is slightly different from the production version shown in Figure 5.

sible for the PC software to be waiting for reply while the query is still stuck in the buffer. If errors occur when downloading programs into the flash memory of the microcontroller, reducing the size of the buffer may improve the situation.



Board assembly

The printed circuit board (Figure 5) is also available fully assembled so readers with relatively little soldering experience don't have to worry about problems with the Prolific IC (IC1), which is only available in a 28-lead SMD package (SSOP-28). If you assemble the circuit board yourself, you should start with this IC because it's easier to solder if the board is otherwise empty.

Numerous *Elektor Electronics* SMD projects have shown that they can be soldered without using special SMD tools. Start by soldering two diagonally opposite leads in place, after first carefully positioning the IC. A check with a loupe will show whether you have a steady hand. If necessary, you can still adjust the position relatively easily at this stage. Continue by soldering an entire row of leads with a sufficient amount of solder. Any excess solder can be easily removed afterwards with a length of solder wick. If your final inspection with a loupe does not reveal any solder bridges, everything is OK. The author has also found that strong reading glasses (3 dioptre) are an excellent alternative to a loupe. They're almost as good as stereo microscope, at only a fraction of the price.

The remaining components shouldn't present any special difficulties. You can omit the pin headers (K5 and K8) if desired. If you wish, you can later connect flat cable here or wire only the connections you actually need to the prototyping area. A socket header has proven to work well as an LCD connector. The matching LCD module can then be fitted with long pin headers and simply plugged into the board.

Initial test

Before using the board for the first time, you should re-inspect the component assembly and solder joints and fit the necessary jumpers. The following jumper settings are important:

JP 11: toward K7/K8 ('ext' position, with power coming from an external supply)
JP 3–6: fitted (LEDs connected)
JP 12: fitted ('Power' LED connected)
JP 2: fitted (debug mode)
JP 7–10: toward IC3 (both interfaces via RS232)

You should check the supply voltage on the board before plugging in the R8C/13 daughterboard, in order to avoid any risk to the microcontroller. LED D6 should light up after you connect an AC adapter with 9-V DC output. The voltage measured on K6 between pin 5 V_{SS} (Pin 5) and pin 7 (V_{CC}) must have a value of +5 V. After verifying this, disconnect the AC adapter and plug in the daughter board while no power is applied to the board.
Now comes the most exciting moment! Reconnect the 9-V AC adapter. The 'Power' LED will light up. Connect K4 to the COM1 serial port of the PC. You can immediately download a program now, because JP2 is configured for debug mode. Use the FDT program as described in the February 2006 issue of *Elektor Electronics*. A good choice for your first download is the 'port_toggle' program. The programming tool will indicate whether the download process completed successfully.
Next, remove jumper JP2 and briefly press the Reset button. The four LEDs connected to Port P1 should start blinking.
If you want to use the USB interface, fit JP7 and JP8 in the proper positions (toward the USB connector). That will connect RXD1 and TXD1 to the USB converter IC. After that, you must configure FDT to use the virtual COM interface. Everything else works exactly the same as with a real RS232 interface.

KD30 debugger

Up to now, we've only described how to download finished programs into the microcontroller and run them. When you're developing a program, especially one that's relatively complex, it's helpful to use a debugger. That allows you to interrupt the program at any desired location, view memory contents, execute single steps, and much more.
The microcontroller must always be in debug mode when you use the debugger, which means that jumper JP2 must remain fitted. The Reset button isn't necessary in this case, because you can only use software commands from the debugger to start and stop your program.
We've prepared an extensive description of how to get started with the debugger and placed it on the Elektor Electronics website at www.elektor-electronics.co.uk. Deeopnding on the amount of feedback we get, a special R8C activity page may also be created, so check out the website.

(050179-3)

References:
www.glyn.com
www.glyn.de
E-mail: elektor-r8c@glyn.de

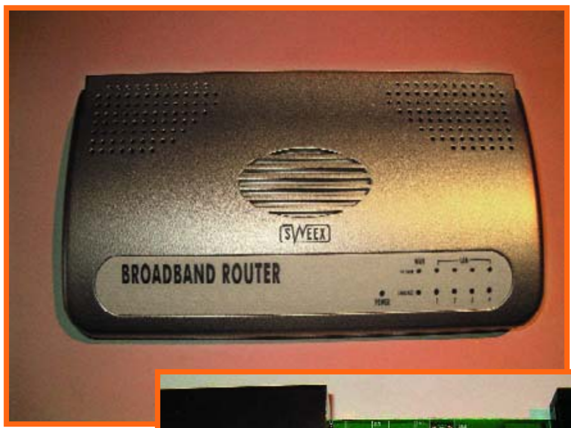
R8C forum:
For hints and tips about using the R8C/13 board, check out the 'R8C 16-bit Micro Starter Kit' topic in our Forum menu at www.elektor-electronics.co.uk. Naturally, we would be pleased if you post your own tips and tricks on this forum for the benefit of other readers.

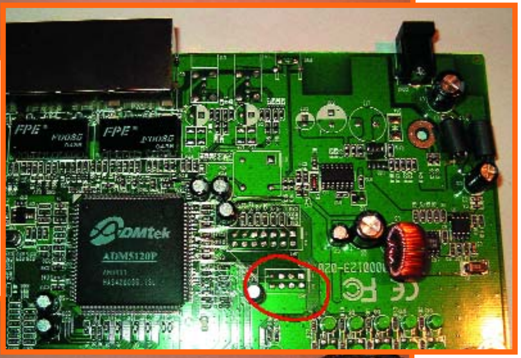
Cheap Logger

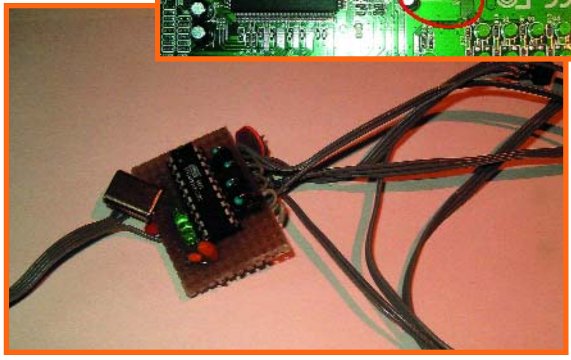
Use a web router to read the gas/water/electricity meters

Jeroen Domburg & Thijs Beckers

This month we'll turn an inexpensive router into a data logger. This is provided with sensors that monitor the readings on gas, water and electricity meters. One of the benefits of using a router is that the logger can be connected directly to an internal computer network and can therefore be accessed from any PC with an Internet browser.







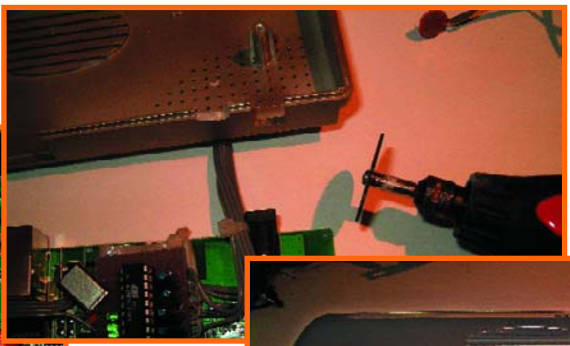
In last month's article we described an inexpensive router and showed how it could be reprogrammed with our own firmware. This month we'll use the same router, with some additional electronics, and build a gas, water and electricity consumption meter.

This meter has four sensor inputs and can therefore read a maximum of four meters. The consumption is read out and stored every five minutes, making it possible to create nice graphs from the data. In this way you can find out during which part of the day the consumption is highest, which can lead to better use of the resources and smaller utility bills.

The device itself can be permanently fixed close to the meters. All measurements can be accessed via the built-in web server, programmed specifically for this task. The meter reader can therefore be hooked up to the home network and accessed from any computer.

The router used is again the 'Sweex broadband router' with four 100 MBit ports, type number LB000021 (without WLAN). The reason we've chosen this router is that it is still one of the cheapest available (£ 17.50 from scan.com). The rest of the hardware isn't pricey either: the costliest are the microcontroller (an ATTiny2313 made by Atmel,

The Sweex LB000021 is still one of the cheapest routers that can run Linux. At the time of writing, this model was available on the Internet for less than £20, including postage costs.
When we look inside this router we see a pair of unpopulated headers on the board. The smaller of the two (circled in red) is the serial port, which we'll use for this project.
This is the board with the microcontroller, which we'll hook up to the port. The circuit is so simple that it can easily be built on a piece of experimenter's board.



The board is connected to the serial port and fixed to the router PCB using a folded piece of duct tape. There is no need to use expensive screws or other fasteners.
A multitool is ideal for making the hole in the case for the sensor cables. A cable-tie around the cables functions as a strain relief.
The router is connected, ready for the firmware update. The network cable connects directly to a PC. This UTP cable may be either a crossover cable or a 1:1 cable.

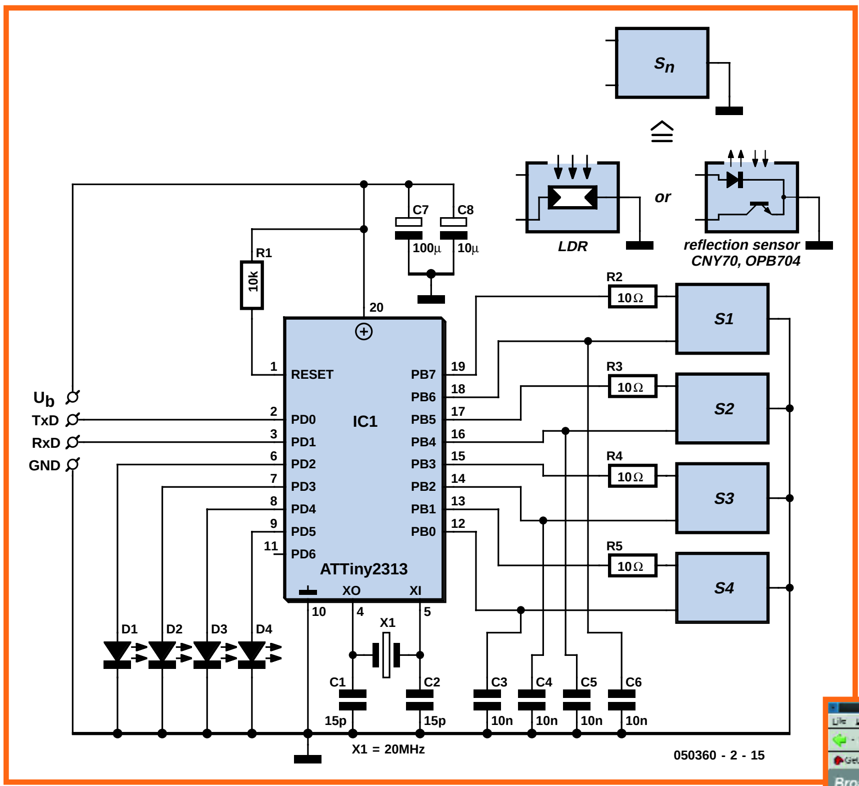


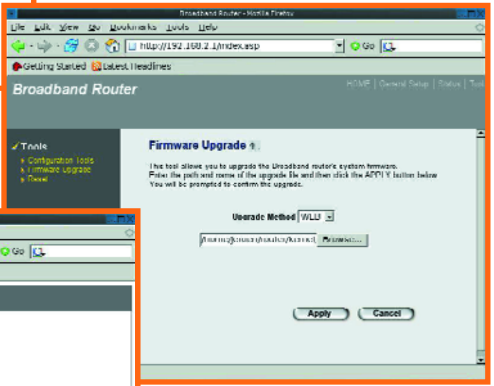
Figure 1. The microcontroller circuit is fairly simple and can easily be built on a piece of experimenter's board (see inset).

which should be available for a few pounds) and the sensors.

This project consists of three parts: the router itself, which acts as web server and stores the data, the board with the microcontroller, and the sensors. The router communicates with the microcontroller every five minutes. The latter uses sensors to keep track of the meters.

The sensors are optical and the types used depend on the meter they have to read. The general idea is that the sensor 'looks' at the gas, water or electricity meter.

Most modern electricity meters have an LED, which flashes every time a certain amount of power has been consumed. This pulse can of course be easily detected by a simple LDR. Older meters have a rotating disc with black

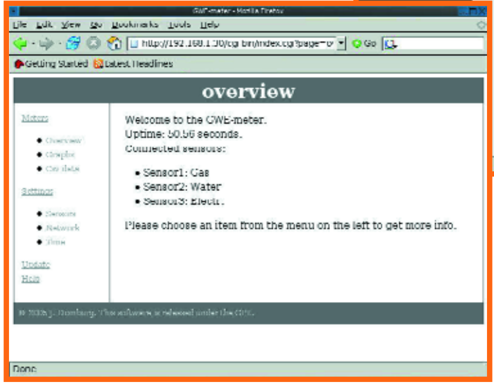


Broadband Router - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://192.168.2.1/index.asp
Go
Getting Started
Latest Headlines
Broadband Router
HOME | General Setup | Status | Tool
Tools
Configuration Tools
Firmware Upgrade
Reset
Apply
Cancel

Firmware Upgrade

This tool allows you to upgrade the Broadband router's system firmware.
Enter the path and name of the upgrade file and then click the APPLY button below
You will be prompted to confirm the upgrade.

Upgrade Method WLU



GWE-meter - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://192.168.1.30/cgi-bin/index.cgi?page=o
Go
Getting Started
Latest Headlines
Meters
Overview
Graphs
Get data
Settings
Sensors
Network
Time
Update
Help
Done

overview

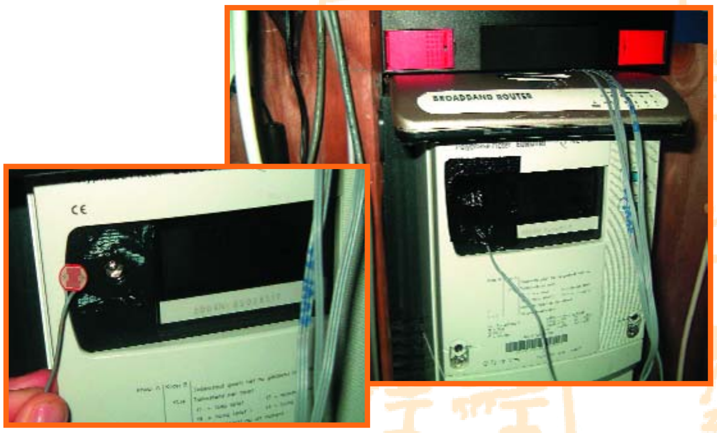
Welcome to the GWE-meter.
Uptime: 50.56 seconds.
Connected sensors:

- Sensor1: Gas
- Sensor2: Water
- Sensor3: Electr.

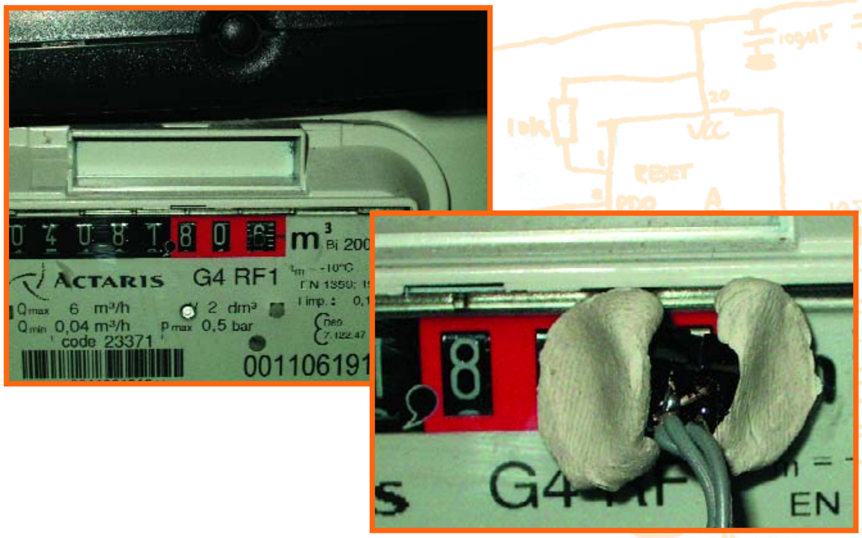
Please choose an item from the menu on the left to get more info.

© 2005 J. Domburg. This software is released under the GPL.

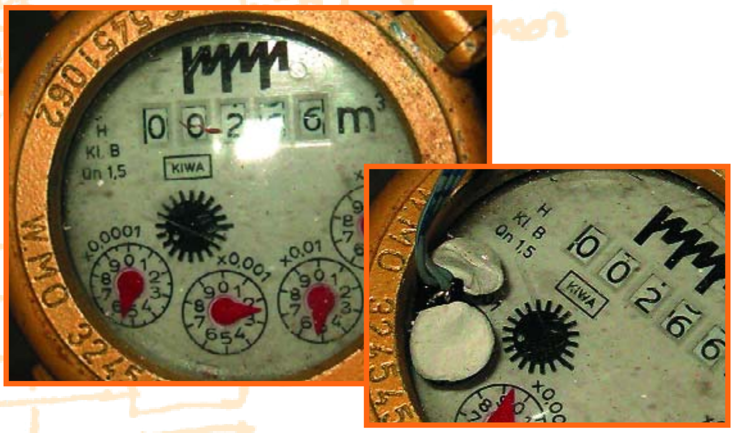
Once everything is connected, the router can be found at IP address 192.168.2.1. The firmware can be upgraded to the GWE-meter firmware by selecting 'firmware upgrade' from the 'tools' menu.
It works! With the firmware upgraded, the router can be connected to the network. The ex-router will be given an IP address automatically. Take a look at the configuration of your gateway, router or cable/ADSL modem to find out what the address is.


The time has come to connect the sensors. A modern electricity meter with a flashing LED is the easiest... An LDR covered by a piece of duct tape is sufficient to obtain an accurate reading.

markers. These meters can be read with the help of reflective optical sensors such as the CNY70, or the OPB704 (more expensive, but better).
Gas and water meters usually have a mark on one of the least significant digits (often the '0' or '6'). A reflective sensor can be used without too many problems in these cases as well.
Whichever type of method is used for the measurements, all sensors occasionally exhibit a drop in their resistance. The capacitors connected to the sensors (see **Figure 1**) and some clever firmware in the microcontroller enable these resistance changes to be read.
The firmware keeps track of the number of times that the current drops below a certain value and hence how often a change of resistance has occurred. This count gives a good indication of the consumption over a given time. The counters can be read or reset via the serial port of the microcontroller. The calibration of the sensors also takes place via this route.
The microcontroller's serial port is connected to a header in the router (JP2). This header normally isn't mounted on the board. The serial port of the router ends up at this header (see **Figure 2**). The router itself runs a dedicated version of the firmware. The firmware contains a program that takes care of the communications with the microcontroller and makes sure that the counters are read at the right time. As we mentioned earlier, this occurs every five minutes. The data are stored in the internal RAM of the router. This has the advantage that about a year's worth of data can be stored; the disadvantage is that all this data will be lost if there is a power cut. Luckily, this can easily be prevented by the inclusion of a back-up battery in the power supply; this will be covered in detail later on.
The firmware inside the router contains a simple web server and several server-side scripts; these enable everything, including the network settings and sensor calibrations, to be accessed via a browser. It is also possible to use the browser to view the data in different formats. The data of one or more selected sensors can be viewed as a graph. Alternatively, the raw data can be downloaded as a CSV file, for use in Excel.
The facility to update the firmware hasn't been forgotten either. Using this update facility, the official Sweex firmware can be flashed back into the router, so it can be used again as a normal router. It's worth noting that the meter firmware hasn't stopped the internal switch from working: the four LAN ports and the WAN port are configured as a single switch by this firmware.
So far so good, but how can you obtain this firmware? The answer is very simple: the required firmware is available from both the Elektor website [1] (file number **050360-11.zip**), and the author's own site [2]. The Linux kernel and all other tools used are licensed under the GPL (General Public License), which means that the source code of the programs has to be made available along with the firmware. The author has of course complied with this requirement. He has also made the tools available that were written specifically for use with the meter firmware. These can also be downloaded from the above-mentioned sites.
Installing the firmware is fairly simple: the router already has a facility to upload a firmware upgrade and this method can also be used to smuggle our firmware into the router; refer to the inset for more info. If you want to flash the original firmware back into the router, you can use the method



The circle of the '6' in this gas meter is not filled in, which can be detected by a reflective optical sensor.
The reflective sensor is fixed to the meter using a bit of Blu Tack.



This water meter is a lot more difficult to read: there are no distinguishable digits, no flashing LEDs...
In principle it should be possible to use a reflective sensor to detect a dial, possibly using a blue LED. However, this has not been extensively tested by the author.

described earlier.
The microcontroller can be programmed via the parallel port of a PC and a programmer [3], consisting of a DB25 connector and three resistors. The required programming software is available for Windows as well as other operating systems.
The physical placement of the sensors depends very much on the type of meter they're used with. A meter with a flashing LED is easiest: an LDR can be stuck with tape over the LED. A gas or water meter can be trickier, but as long as the reflective sensor can point straight onto the last digit, it shouldn't cause too many problems either. The author couldn't test an electricity meter without an LED (i.e., one with just a rotating disc), but it is likely that the black marker on the disc can be detected by a reflective sensor after a bit of experimentation.
Once the sensors have been fixed into place and connected to the microcontroller, they have to be calibrated. There is a link on the web page of the router that helps with this. This function continuously reads the (analogue) value of the sensor until the user has determined that the calibration is complete. The minimum and maximum signal levels are extracted from these readings. Each time that a sensor's output changes by more than two-thirds of the difference between minimum and maximum, the microcontroller assumes that another unit of gas, water or electricity has been used.
There is no need to worry that the router itself will significantly increase your electricity bill: it only draws about 340 mA from its 12 V supply. The power consumption is therefore about 5 watts. The router itself contains a switch-mode PSU; any voltage from 6 to 15 Volts was found to be suitable.
Armed with this information, we can design a simple back-up circuit to keep the data safe during a power cut. A 9 V NiMH or NiCd battery can keep the router powered for some 20 minutes. Six beefy NiMH cells give a back-up time of about three hours. During this period the meter continues to take readings. The measurements of all connected meters are therefore preserved.
The circuit in **Figure 3** shows how a back-up supply should be connected. This circuit should be inserted between the adapter and the router. The two diodes ensure that no current can flow from the adapter to the battery and vice versa. The resistor provides enough current to the battery to keep it fully charged during normal use. If you use Schottky diodes the voltage drop, and hence the losses, are kept to a minimum. The value for the resistor is calculated as follows:

R =
$(U_b - U_{batt})$ / (0.02 × battery capacity)

For example, when a 12-V adapter and 9-V battery with a capacity of 800 mAh are used, the resistor should take a value of about 180 Ω.

(050360-2)

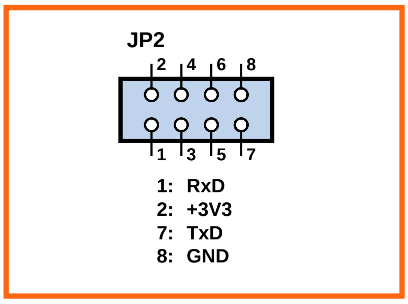


Figure 2. These are the connections of JP2 in the router.

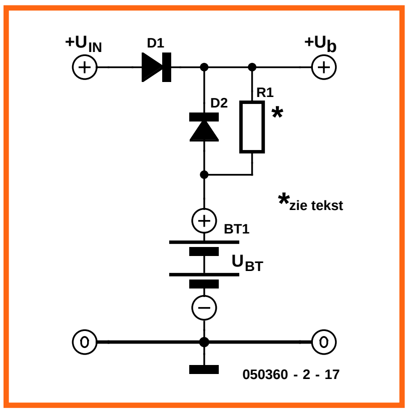


Figure 3. The back-up supply is inserted into the supply line. This diagram shows that it is a fairly simple circuit.

Web links

[1] www.elektor-electronics.co.uk/, items March 2006.

[2] http://sprite.student.utwente.nl/~jeroen/projects/gwemeter

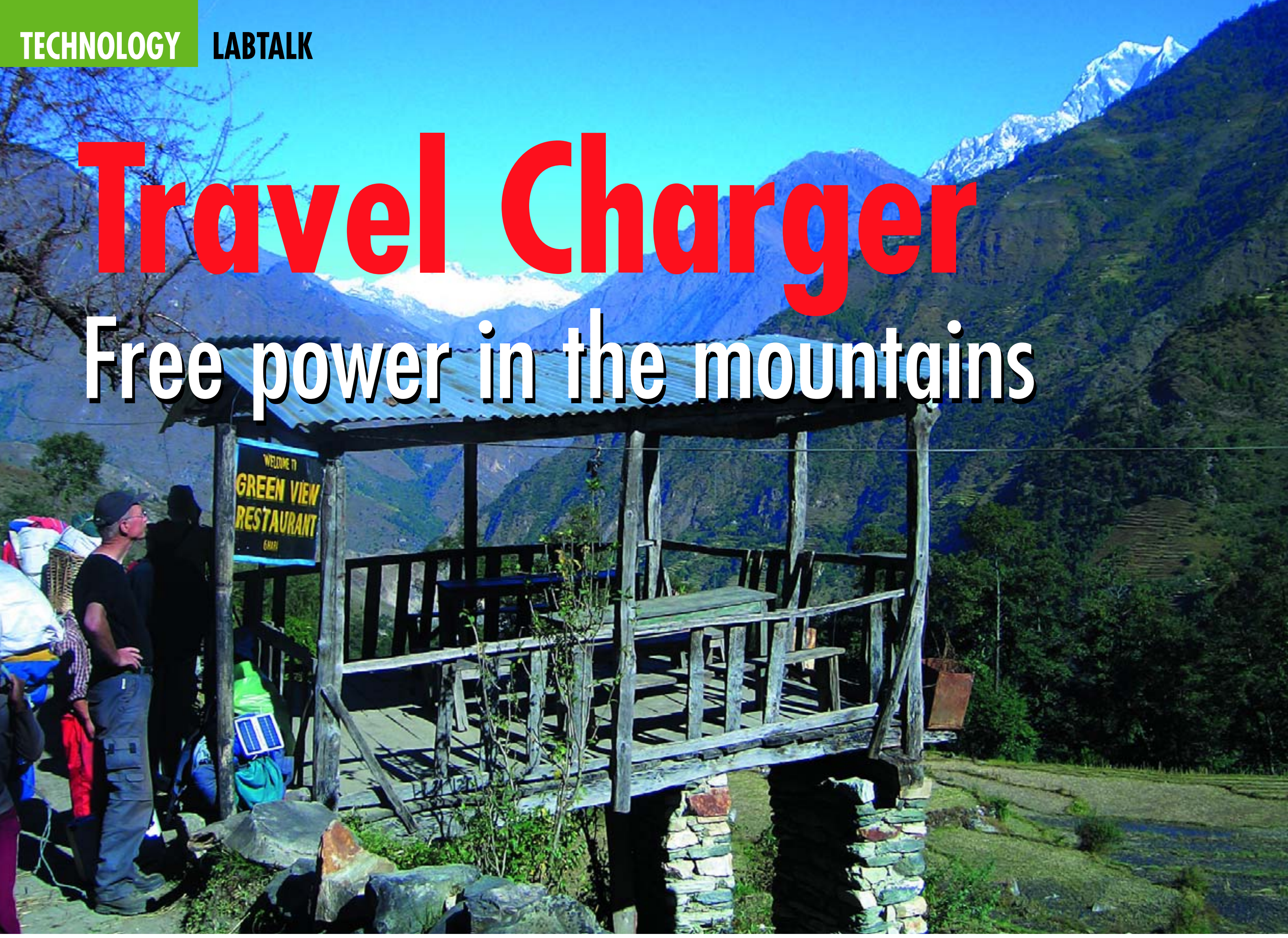
[3] http://sprite.student.utwente.nl/~jeroen/projects/progavr

About the author:

Jeroen Domburg is currently studying electronic engineering at the Saxion University in Enschede (The Netherlands). He is an enthusiastic hobbyist, who spends much of his spare time on microcontrollers, electronics and computers. In this section we will be able to examine and build some of his projects.

Travel Charger

Free power in the mountains



Karel Walraven

Recharging batteries for devices such as digital cameras can be a problem when you're on holiday in a remote or inhospitable region. You won't find an electrical outlet anywhere. Naturally, you can count on an *Elektor Electronics* designer to whip up a solution using a solar panel and a DIY Li-ion battery charger.

I was due a bit of a holiday, so I suggested to management that I'd step out for half a year. Their response was less than enthusiastic, and my own enthusiasm was soon dampened when I had a good look at what it would cost. The outcome was thus four weeks of trekking through the Annapurna region in Nepal.

To meet my electrical energy needs, I took along a solar panel and a simple Li-ion charger built from discrete components. I didn't have a lot of time, and in the spirit of the motto 'better a good copy than a bad design', I cribbed the accompanying circuit from a well-known Chinese company.

As you doubtless know, charging lithium cells is actually quite easy. You generate a well-regulated voltage of exactly 4.1 or 4.2 V (always read the manufacturer's specifications to determine the right value) and add current limiting to keep the current within bounds. The charging current will automatically decrease as the cell becomes charged (see **Figure 1** for the charging characteristic).

You can assume that the cell is fully charged when the charging current drops to 1/20C or less. Just to reiterate, the charging voltage is critical – the allowed tolerance is only 1%. That's not very much, because 1% of 4.2 V is only 42 mV. That means you have to measure the output voltage carefully to ensure that it stays within tolerance.

The nice thing about this circuit (**Figure 2**) is that it's easy to build as a DIY project because it doesn't use any obscure components. The TL431 voltage reference is an old standby that you can obtain almost everywhere. For the rest, it consists of a few ordinary transistors and a power transistor, all of which can be replaced by any reasonable equivalents. The Schottky diode can be any type that can handle 1 A, or if necessary you can use an ordinary 1N4001. The input voltage can also be higher, although beyond a certain point you'll have to fit a heat sink for T1.

Design philosophy

It's always interesting to examine someone else's design carefully in order to figure out the underlying design philosophy.

Reference voltage

To start with, the circuit is based on a good, stable reference voltage generated by a Texas Instruments TL431 (IC2). That's a prudent choice, because a good reference is a fundamental requirement if you want to be sure of achieving 1% accuracy for the charging voltage.

The reference IC is followed by an emitter follower (T4) so the reference source can supply more than just a few milliampères. That would normally reduce the output voltage by approximately 0.6 V and trash the stability, but feedback from the transistor via R21 and R23 eliminates those problems.

The IC adjusts its reference potential to maintain a voltage of 2.5 V on its Adjust input. That yields a reference voltage of 3.3 V, which is used everywhere in the circuit. It's important to be able to adjust the value of R21 or R23 when you're building charger if the voltage of the TL431 differs too much from the nominal value. If a check reveals that your reference IC is near the end of the tolerance range, correct the reference voltage by fitting a resistor in parallel with R21 or R23. You can simply use the trial-and-error method by trying a succession of values until you obtain the correct voltage. Once that's done, you can solder the resistor permanently in place.

Naturally, a reference voltage of 3.3 V represents an arbitrary choice. It could also be a bit higher or a bit lower. It shouldn't be all too high, though, to avoid restricting the headroom of the TL431. A more important consideration is that the reference voltage must fall within the common-mode input voltage range of the opamp used in the circuit. We'll say more about that when we talk about selecting the opamp.

A decidedly low reference voltage, such as 2.5 V (as obtained without a voltage divider), is also undesirable because it makes it more difficult to drive LED D1 and increases the sensitivity of the circuit to the input offset voltage of the opamp.

Control circuitry

Now we come to the actual control circuitry. We want to have:

- a stable output voltage
- current limiting
- battery monitoring (for temperature)
- a clear indication when the battery is fully charged

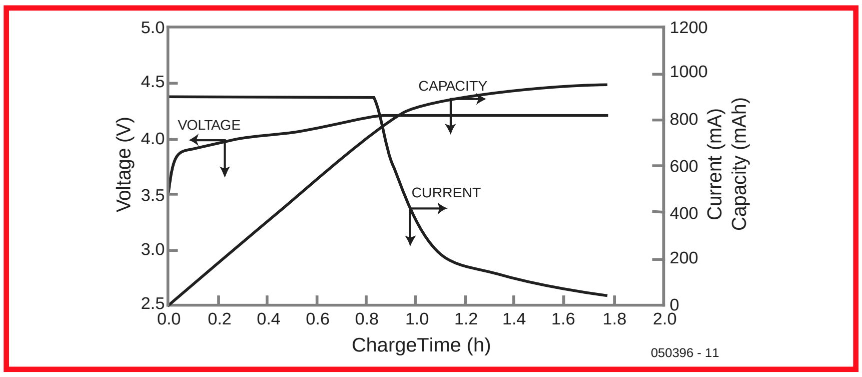


Figure 1. Charging characteristic of an Li-ion cell (source: Panasonic).

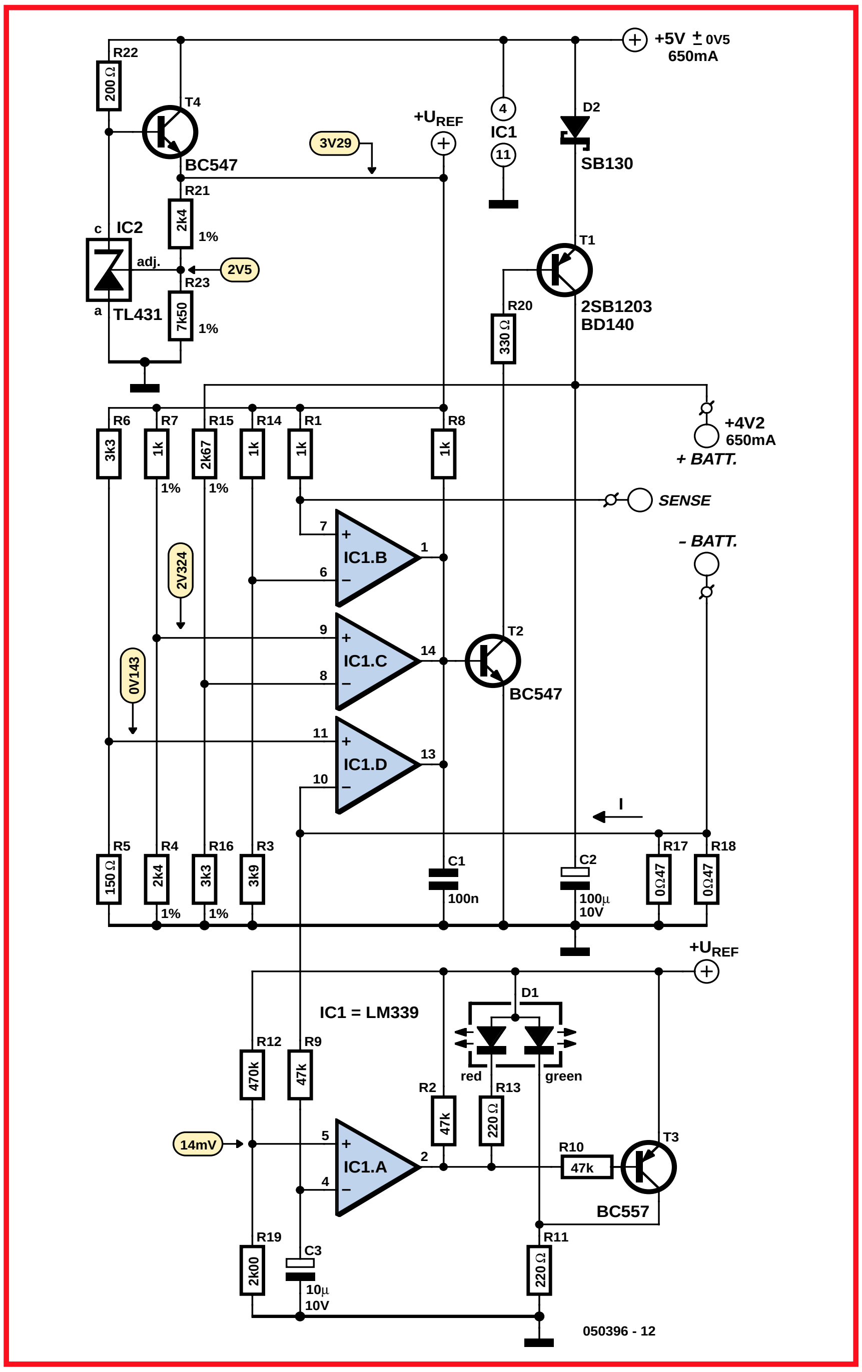


Figure 2. The circuit of the charger uses standard components instead of special charger ICs.

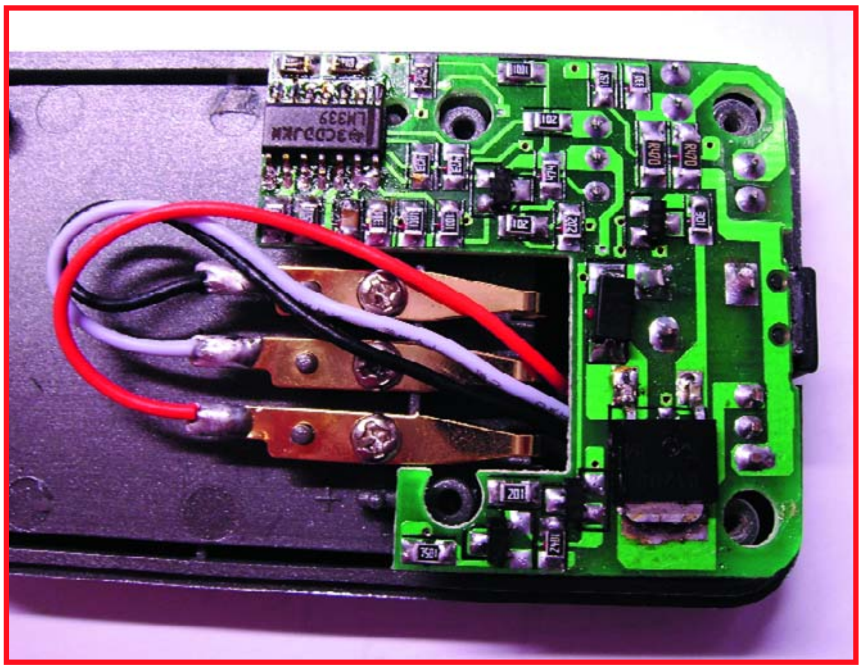


Figure 3. The small printed circuit board for the charging circuit. The spring contacts for the battery pack are at the right.

The first three items all affect the output voltage. The output current can be reduced by lowering the output voltage, and if the battery becomes hot, the charging current must be reduced by lowering the output voltage.
Here we have a topology in which three opamps (IC1b, IC1c and IC1d) can simultaneously control a single output. That's possible using a sort of OR-gate arrangement. In the schematic diagram, you can see that the outputs are simply connected together. Under normal conditions, that would naturally lead to problems whenever one opamp tries to increase the output voltage while another one tries to reduce it. That problem can be solved by putting diodes or transistors after the opamps.
In this case, a more elegant solution is to use opamps with open-collector outputs. That means each opamp can only *reduce* the voltage on the shared output. Resistor R8 provides the pull-up voltage. It supplies the base current for T2, which in turn drives T1 into conduction and voilà, the positive terminal of the battery is connected to the positive supply voltage. This gives us a system in which each parameter (voltage, current, and temperature) has its own opamp that can reduce the output voltage.

Current regulation
As a rule of thumb, you can assume that the maximum charging current of a normal Li-ion battery is around 0.7 to 1 times its capacity. This charger delivers a maximum current of 0.65 A, which makes it suitable for batteries rated at 0.65 Ah or more.
The current with a fully discharged battery would ordinarily be higher than the above-mentioned 0.65 A. The current can be easily measured by connecting a small resistance in series. Here that role is fulfilled by R17 and R18, with two resistors being used due to the magnitude of the current and the power dissipation. The current generates a voltage across these resistors, and IC1d compares this voltage with a reduced reference voltage obtained from a voltage divider. If the voltage generated by the charging current is higher than the comparison voltage, IC1d takes control by reducing the voltage on the shared output. That lowers the charging voltage for the battery and thus reduces the charging current to the desired level.
When the battery becomes fully charged, the control function provided by IC1c prevents the voltage from rising above 4.2 V. The battery voltage is compared with the reference voltage via two voltage dividers (R15/R16 and R7/R4, respectively), and if necessary IC1c reduces the voltage on the shared output.
The logic of the temperature monitoring circuitry with IC1b is similar. A thermistor (NTC resistor) is built into the battery, and its resistance decreases as the temperature increases. The thermistor forms a voltage divider in combination with R1, and IC1b compares the voltage at the junction of this divider with a reduced reference voltage obtained from divider R14/R3. It restricts the charging process if the temperature rises too high.
By the way, this temperature protection is not fail-safe, so charging will proceed as normal if there is bad contact or an open lead.
If you've ever built a circuit with an opamp followed by two transistors (which also provide gain), you know you can expect problems. Such a circuit is guaranteed to oscillate. That problem is solved here by using a hefty capacitance (C1) to slow down the entire control loop. An additional capacitor (C2) connected at the output, in combination with voltage divider R15/R16, keeps the circuit stable in the no-load state, and the divider provides a minimum load.

Indicator
We haven't said anything yet about the indicator portion of the circuit, which consists of the circuitry around IC1a. This bit of electronics is not essential to the operation of the circuit, but it is nevertheless very important. As a user, you naturally want to know where things stand. The nice thing about this indicator is that it responds to the charging *current*. That means you can be sure that charging is actually taking place, which avoids simple errors such as forgetting to connect the mains adapter or a bad battery contact.
Here again the reference voltage is divided down, this time by R12/R19, to a voltage of approximately 14 mV. If the voltage drop resulting from the charging current through R17/R18 is less than this value (with no battery connected or a fully charged battery), the internal open-collector output transistor of IC1a will be cut off. The red LED in D1 will be off in that case, but the green one will be on because it is connected to 3.3 V and to ground via a 220-Ω resistor.

In the opposite case, with sufficient charging current flowing, IC1a will pull its output low. In that case LED will be on and T3 will conduct to short out the green LED and keep it dark. Using a bi-colour LED for D1 thus provides a perfectly clear charging indicator.
Have you already realised the significance of selecting 14 mV as the refer-

ence voltage for IC1a? The maximum charging current of 650 mA generates a voltage drop of 153 mV across R17/R18. 14 mV is approximately 1/11 of this value, so the battery is considered to be fully charged as soon as the current drops to 1/11 of the maximum value. The charging process will continue as usual, but the LED will indicate that the battery is fully charged.
You could also add another stage to the circuit, because it's better to limit the charging current to 1/10 of the battery capacity if the battery has been discharged to below 2.9 V. That can be easily implemented by using another opamp to connect a resistor in parallel with R5 to reduce the divided reference voltage for current limiting to 14 mV. However, the manufacturer decided not to do this, presumably because it would require an additional IC.

Component selection

Now for a few remarks about selecting the opamp. When you design a circuit, you can choose from hundreds of IC types. All the different options don't make the choice any easier. However, in this case a standard opamp type without any special properties is sufficient. It doesn't need to have especially high accuracy (a millivolt more or less doesn't matter), it doesn't have to have a special temperature range, and noise and speed are also unimportant. However, there are two aspects that deserve further attention.

The first is the common-mode range. When you select an opamp for a circuit, it's important to keep all the voltages that may be present on its inputs within the limits of the common-mode range, or to select an opamp that can handle the expected range of voltages. The data sheet for the LM399 says that the lower limit is 0 V and the upper limit is 1.5 V below the supply voltage. If we had chosen an LM741, the lower limit would have been 1.5 V, so a voltage such as 14 mV would not have been allowed. That means it's essential to chose an opamp for this design that can work properly with voltages close to zero.

The second aspect is the offset voltage. From the data sheet, you can see that the input offset voltage is 2 mV. (The manufacturer supplies this opamp in various versions with different offset voltage ratings. As you might expect, better specs always cost more.) That means the IC introduces an error of approximately 2 mV at its inputs. As a result, although you think (or hope) that the charging indicator will change state at 14 mV, it may actually change as early as 16 mV or not until 12 mV. That's an error of more than 10%, which is considerable. That isn't a problem for the charging indicator, but such an error in the charging voltage would be unacceptable. That's why the reference value for the charging voltage is much higher at around 2.4 V. An error of 2 mV at 2325 mV is only around 1 part in a thousand, which is negligible.

(050396-1)

Construction Guidelines

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

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

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

In some circuit diagrams, to avoid confusion, but contrary to IEC and BS recommandations, the value of components is given by substituting the relevant prefix for the decimal point. For example,
3k9 = 3.9 kΩ   4μ7 = 4.7 μF

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

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

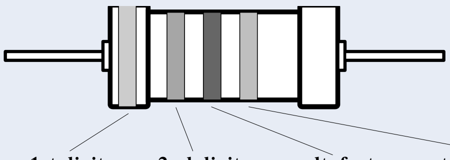
Soldering. Use a 15–30 W soldering iron with a fine tip and tin with a resin core (60/40) Insert the terminals of components in the board, bend them slightly, cut them short, and solder: wait 1–2 seconds for the tin to flow smoothly and remove the iron. Do not overheat, particularly when soldering ICS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

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

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

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

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



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

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

A real µC PLC using an AT89S8252 board

Erik Martens

What turns a mC board into a real PLC? A PLC is a controller with a program stored in memory, where reliability, availability and safety are of the utmost importance. Only when a mC board can be operated as a PLC and it complies with industrial standards can it be called a real PLC.

This project covers all aspects to turn a standard AT89S8252 µC board into a real PLC, using a minimum of off-the-shelf components. To add the PLC functionality to an AT89S8252 board we require a small section of extra software (firmware) in the microcontroller. Your own µC PLC program then works under supervision of this firmware.

System overview

To turn a standard AT89S8252 board into a PLC we make use of four I/O pins on the microcontroller (P1.0 to P1.3). When the controller is reset it knows from the state of the flip-flop (IC2a and IC2b, read via P1.0) whether a hardware reset or an internal reset occurred. After a hardware reset the firmware sets the flip-flop using P1.1. Port P1.1 also turns off all output sections (a high level from P1.1 turns off T1 via IC3d and IC2d).

The STOP/RUN/RESET switch (S1) is used by the firmware to start and stop the PLC program. This switch can also be used (via R3) to force a hardware reset of the µC. P1.2 is used to read the state of the STOP/RUN/RESET switch. When the switch is in the STOP position, all outputs are automatically turned off. LEDs D7 and D8 show the current status of the PLC program.

Should the level of the supply voltage to voltage regulator IC4 drop below about 8.2 V, it will be reported to the µC via P1.3, after which all output modules are immediately turned off.

Two input modules have been included in the circuit diagram, but you can add as many as you need to ports P0, P2 or P3. The inputs are electrically isolated, can take $+24V_{DC}$ signals, are protected against voltage spikes and polarity reversal.
The output sections are also electrically isolated. You can connect as many output modules as you need to ports P0, P2 or P3. However, the output sections can only be active when T1 conducts. This will only happen when all operating conditions are satisfied (P1.1 and P1.2 high, STOP/RUN/RESET switch in the RUN position and a high enough supply voltage).

To program the µC you can use the SPI port (P1.5, P1.6 and P1.7) on the µC. The programming can only take place when the reset line is active. For this reason R8-R11 and IC3a have been added, so that a high RS232 input (for example DTR) will keep the µC reset line permanently high. The programming functionality via the SPI port is not shown in the circuit diagram.

Firmware

The main PLC program works under supervision of the so-called firmware. An example of this firmware has been included in this article and you will find it to be relatively small (about 100 bytes). The µC PLC firmware was written using the TASM shareware assembler. The use of this assembler has been covered in previous Elektor articles: '89S8252 Flash Microcontroller Board' (Dec. 2001), 'Microcontroller Basics Course' parts 1 and 2 (Jan./Feb. 2002). These articles also describe the different ways in which the microcontroller software can be put into the microcontrollers (external programmer, SPI).

The firmware takes care of the following functions:

Reset
After a reset the µC will immediately start running the firmware. If a hardware reset occurred, the PLC will start the first PLC cycle if all other conditions are satisfied (STOP/RUN/RESET switch in the RUN position and a good supply voltage). When an internal reset occurs (when the cycle time was exceeded), the STOP/RUN/RESET switch first needs to be set to the STOP position, and then to the RUN position, before the PLC cycle will start again

Please note: this project is has not been tested or post-engineered by the Elektor Electronics laboratory.

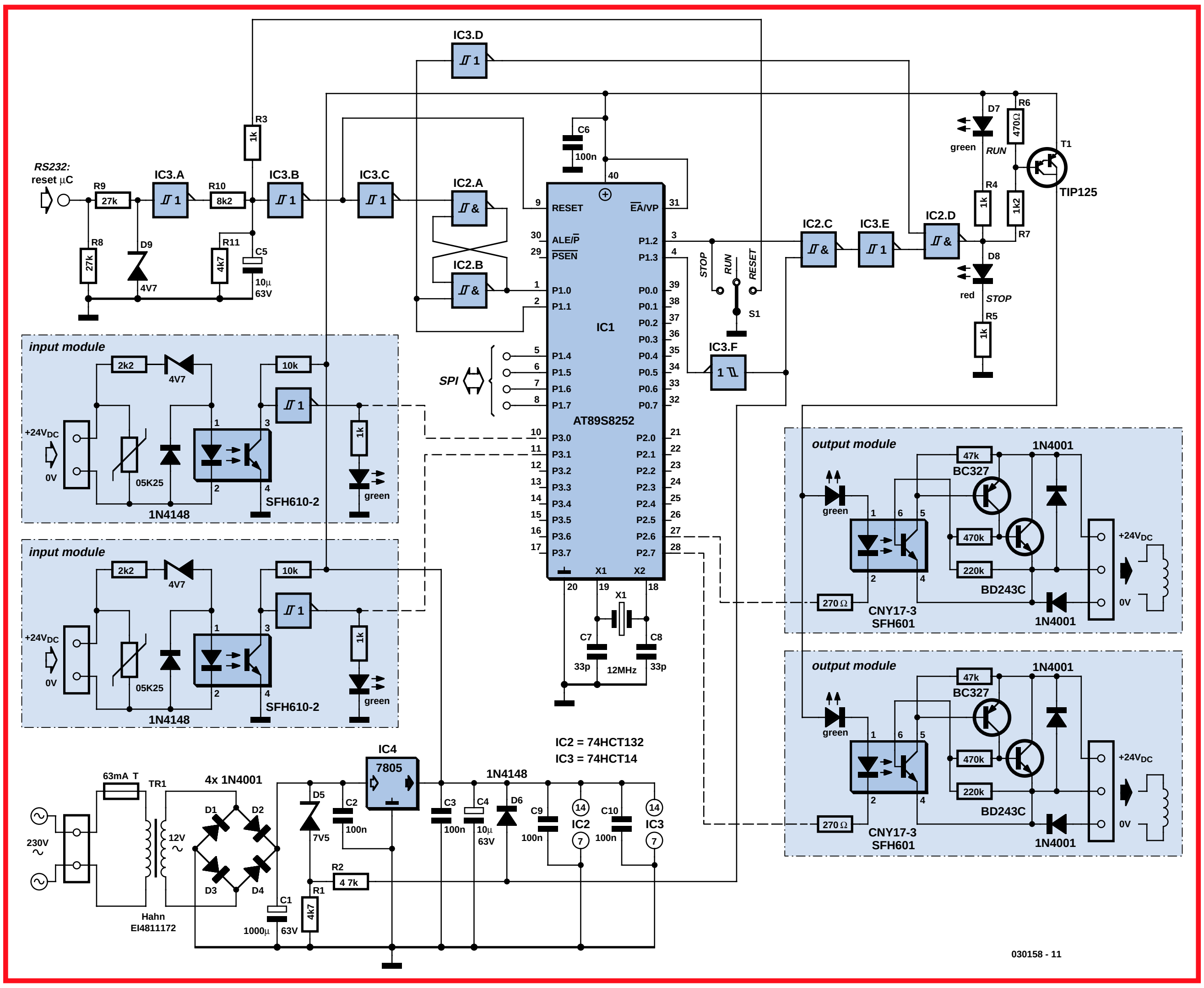


Figure 1. Circuit diagram of a real PLC using an AT89S8252.

RUN-STOP function

The most important task of the firmware is to make sure that all conditions are satisfied before starting a PLC cycle. A PLC cycle can be started if the following conditions are met:

- the STOP/RUN/RESET switch is in the RUN position;
- the supply voltage is high enough;
- the STOP/RUN/RESET has been switched from RUN-STOP-RUN after the PLC cycle time has been exceeded.

STOP/RUN/RESET switch

This switch is used to control the running of the PLC program. When the switch is turned to the STOP position the currently running cycle is still completed, after which no new PLC cycle is started. When the PLC is in the STOP condition all output modules are turned off.

Supply voltage monitoring

Before the start of every PLC cycle the firmware checks that the supply voltage is (still) in order. Should the supply voltage at the input of voltage regulator IC4 drop below about 8.2 V, a new PLC cycle will not begin and the ERRPS routine will be called instead. This routine can be used to store variables into a non-volatile memory (the EEPROM in the µC) before the supply voltage drops too far for the µC to stop functioning. If there is only a temporary drop in voltage (caused by interference, for example), which doesn't reset the µC, the PLC cycle will start again as soon as the supply voltage is restored.

PLC cycle-time watchdog

You may know that the AT89S8252 has an on-chip watchdog timer. This timer is used to keep an eye on the cycle time of the PLC program. At the start of every PLC cycle the watchdog timer is reset. Should a PLC cycle take too long, for example caused by an endless loop within the PLC program, the internal watchdog will generate an internal reset of the µC. In the reset state of the µC P1.1 will immediately become 'high', which causes all output modules to be turned off. The firmware will then detect that an internal reset occurred (by the state of flip-flop IC2a/b) and will enter the STOP condition. Despite the fact that the STOP/RUN/RESET switch is in the RUN position, the PLC program will remain in the STOP condition. The program only exits from this condition when the STOP/RUN/RESET switch is turned to STOP. This behaviour prevents the PLC from starting again immediately after the PLC cycle time was exceeded. The maximum PLC cycle time can be set to one of the following values: 16, 32, 64, 128, 256, 512, 1024 or 2048 ms.

Using the µC PLC firmware

The PLC firmware included with this article should be used as the basis for your own PLC application. You only need to add your code to the MAIN, INIUC, INIPLC, ERRPS and ERRCY routines to provide the functionality you require.

MAIN: This routine is called every cycle. This is where the inputs are read, the logical relationships are calculated and the outputs are set. The MAIN routine may not run longer than the set watchdog time, so take care with delay loops and 'jumping back'.

INIUC: When power is first applied to the µC PLC this routine is called once. You could for example use this routine to restore variables from the EEPROM of the microcontroller. The example below shows how to get the contents of EEPROM address 16 and store it in the accumulator (see Listing 1).

INIPLC: This routine is called once when the STOP/RUN/RESET switch is set to the RUN position (warm start). You could use it to initialise certain variables with a starting value. You could also initialise a communication link here. The cycle time is not monitored while this routine is running.

ERRPS: The ERRPS routine is called once when the supply voltage is turned off or when it falls below a critical level. In this routine you could store permanent variables (these are variables that have to keep their value during a power failure) into the microcontroller EEPROM. The following example shows how the value of the B register is stored in the microcontroller EEPROM at address 16 (see Listing 2).

ERRCY: This routine is called continuously when the cycle time has been exceeded and the STOP/RUN/RESET switch is in the RUN position. This routine could be used to light an LED that indicates that a program error has occurred. This LED should be extinguished again in the INIPLC routine.

Listing 1

```
ORL     WMCON,00001000B  set EEMEN, select uC EEPROM
MOV     DPL,#010H
MOV     DPH,#000H
MOVX    A,@DPTR          store contents of address 16 into the accumulator
```

Listing 2

```
        MOV     DPL,#010H
        MOV     DPH,#000H
        ORL     WMCON,#00010000B   set EEMWE, write-enable uC EEPROM
M001    MOX     A,WMCON            wait until the uC EEPROM is ready
        JNB     ACC.1,M001
        MOV     A,B
        MOVX    @DPTR,A
        ANL     WMCON,#11101111B   reset EEMWE, write-disable uC EEPROM
```

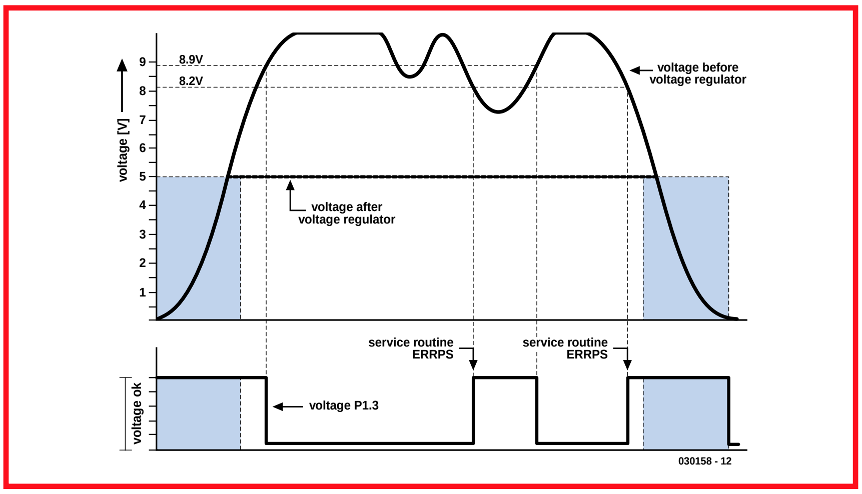


Figure 2. Monitoring the supply voltage.

And finally...

This project forms the basis of a real PLC, where you are free to add a large number of input and output modules according to your requirements. It also complies with industrial standards for signal interfacing, operation and control.

The source code for the controller firmware has been made freely available to Elektor Electronics readers by the author. This file can be downloaded from our website under no. **030158-11**.

(030158)

Attention!

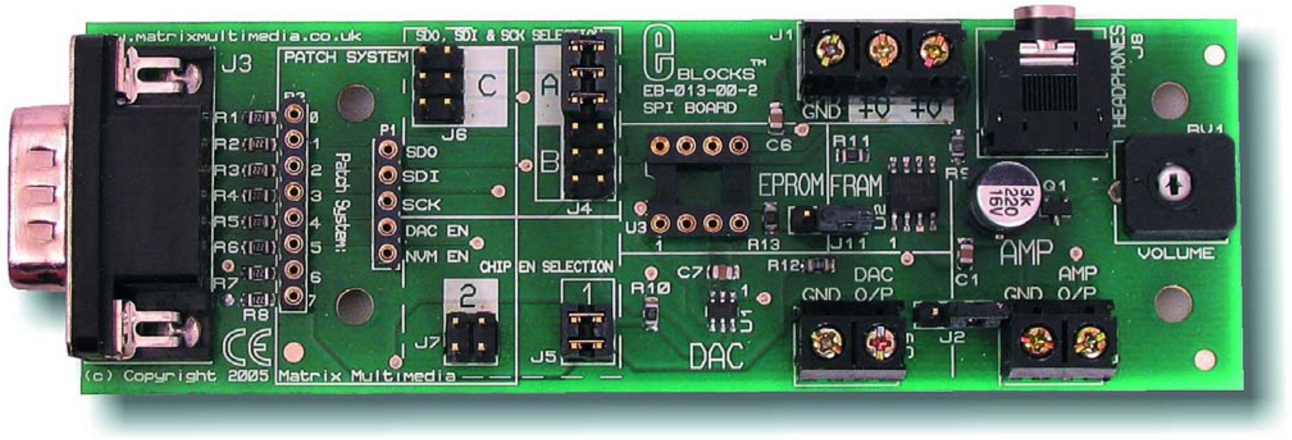
This project uses a transformer that is connected to the mains supply. This could have deadly consequences if you don't comply with all relevant safety standards. For this reason it's important that you follow all instructions shown on the safety guidelines page, regularly printed near the back of the magazine. When you use the transformer recommended in this article you can mount the circuit into a box that complies with the Class-II standard.

Listing 3. An example of the firmware in the 89S8252.

```
;FIRMWARE mC PLC
WMCON       .equ     096H
P1.0        .equ     090H
P1.1        .equ     091H
P1.2        .equ     092H
P1.3        .equ     093H
            .org     0000H                ; start address after a reset
            LJMP     START
            .org     0200H

;RESET mC
START       ANL      WMCON,#11111110B     ; disable watchdog
            JB       P1.0,SOFT            ; jump, software reset

;HARDWARE RESET mC
HARD        CLR      P1.2                 ; disable outputs using P1.2, because P1.1 will be used to set the reset-flipflop
            CLR      P1.1                 ; set the reset-flipflop
            JNB      P1.0,HARD            ; jump, wait until the reset-flipflop is set
            SETB     P1.2                 ; enable outputs using P1.2
            ACALL    INIUC                ; mC initialisation following a reset
            SJMP     NCYC

;SOFTWARE RESET mC
SOFT        ACALL    ERRCY                ; STOP/RUN/RESET switch in RUN position after the cycle-time was exceeded
            JB       P1.2,SOFT            ; jump, wait for the STOP/RUN/RESET switch to be in the STOP position
            ACALL    INIUC                ; mC initialisation following a reset
            SJMP     NCYC                 ; jump, STOP/RUN/RESET switch was set to STOP

;PLC CYCLE
PLC         JNB      P1.2,NCYC            ; jump, STOP/RUN/RESET switch is set to STOP
            JB       P1.3,NCYC            ; jump, faulty supply voltage
            SJMP     WDTR                 ; jump, conditions are met to start a new PLC cycle
NCYC        ANL      WMCON,#11111110B     ; disable watchdog
            SETB     P1.1                 ; disable outputs using P1.1
            JNB      P1.3,PSOK            ; jump, supply voltage OK
            ACALL    ERRPS                ; faulty supply voltage
PSNOK       JB       P1.3,PSNOK           ; jump, wait until the supply voltage is OK
PSOK        MOV      R2,#100              ; delay: #100 x #50 x 2ms = 10.000ms = 10ms
DEL0        MOV      R1,#50               ;
DEL1        DJNZ     R1,DEL1              ;
            DJNZ     R2,DEL0              ;
            JNB      P1.2,NCYC            ; jump, STOP/RUN/RESET switch is in STOP position
            JB       P1.3,NCYC            ; jump, faulty supply voltage
            ACALL    INIPLC               ; PLC initialisation after the STOP/RUN/RESET switch is set to RUN,
                                          ; or when the supply oltage returns
            CLR      P1.1                 ; enable outputs using P1.1, STOP/RUN/RESET switch is set to RUN
                                          ; and the supply voltage is OK
            ORL      WMCON,#11100000B     ; WMCON:    ms          WMCON:  ms
                                          ; #000xxxxx  16         #100xxxxx256
                                          ; #001xxxxx  32         #101xxxxx512
                                          ; #010xxxxx  64         #110xxxxx1024
                                          ; #011xxxxx  128        #111xxxxx2048
            ORL      WMCON,#00000001B     ; enable watchdog
WDTR        ORL      WMCON,#00000010B     ; watchdog-timer reset

;PLC PROGRAM
MAIN        ...
            LJMP     PLC                  ; jump, next PLC cycle

;INITIALISE mC -called once after a hardware reset
INIUC       RET

;INITIALISE PLC (STOP/RUN/RESET switch set to RUN or return of the supply)
INIPLC      RET

;POWER SUPPLY ERROR -called once after the supply fails
ERRPS       RET

;CYCLE TIME ERROR (STOP/RUN/RESET switch set to RUN after the cycle-time is exceeded)
ERRCY       RET
```

E-blocks Making

John Dobson


E-blocks SPI Bus Board containing NVM and DAC

Many of you may be familiar with using the A/D inputs of the PICmicro microcontroller but using D/As is less common. This short article examines how it is possible to use Flowcode and E-blocks to produce a simple sinewave oscillator for use in testing audio circuits.

There is often a requirement for a simple sinewave oscillator you can use for testing audio circuits. The E-blocks system includes a D/A converter (DAC) so why not make a general-purpose waveform generator with it?
To start with, you need to decide on the range of values you want to produce and do some calculations on the overall parameters to check the project is feasible. The D/A has an output range of 0 to 5 V in 255 equal steps. It does not make sense to go too near the supply rails in case some headroom is needed for a buffer device, so to keep things simple, go for a maximum of 4 V peak-to-peak output. This can be divided down in software to get smaller signals. The frequency will need to be around 1 kHz – that's mid-range audio. A quick check of the specification of the MAX5385 D/A states that it has a settling time of 20 µs for an output accuracy of "half of the least significant bit". If we have 256 samples per waveform then this would give us a maximum output frequency of 200 Hz. In theory that's not high enough, but as we are using a sinewave, with small incremental steps between samples, the system has a fighting chance of coping and we can always resort to reducing the number of samples per waveform if it does not perform well.

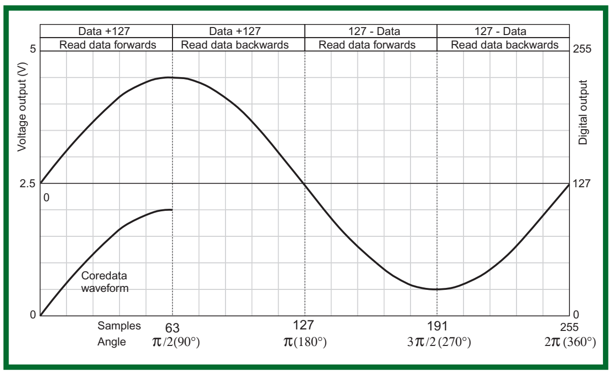


Figure 1. Initial specification sketch.

Get to grips with D/As

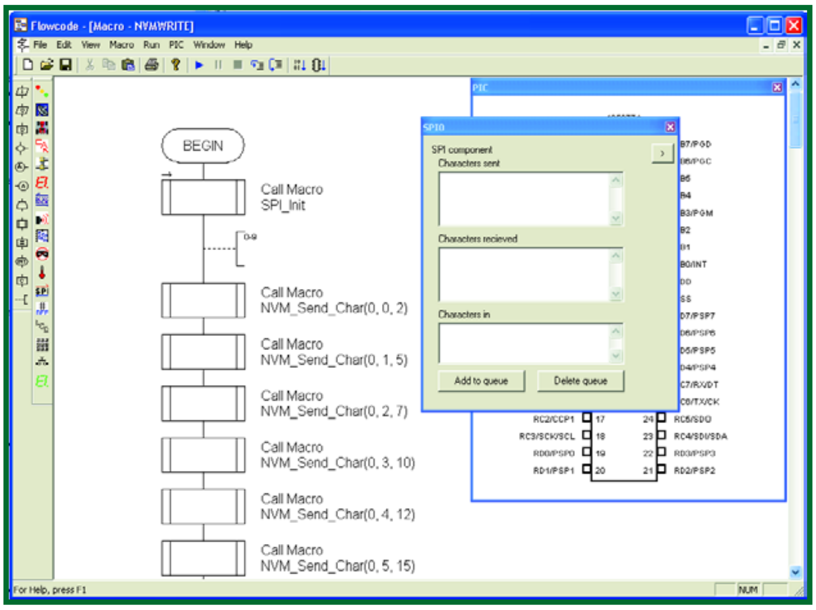
Good news: D/As are relatively simple to use. You feed them a number and the analogue output level equals the full-scale voltage output multiplied by a ratio, *N*, of the number you feed in and the full-scale digital input. From the graph in **Figure 1** you can see an initial sketch assuming 5-V full scale output and eight bits (2^8 = 256 or 0 to 255) input. Actually, the output of the DAC is only 0.9 V_{CC} so there will be a 10% voltage level error in the real output, but we can live with that.
The micro needs to generate a range of values corresponding to a sinewave output. With 256 steps that means 256 separate values. Fortunately, sinewaves are symmetrical as well as periodical so we can actually just

Naves Flowcode in control of DACs

generate numbers for a quarter of a waveform and then use simple programming to reflect the data horizontally and vertically. We want a 4-V peak-to-peak output — to make the numbers easy it's best to actually just use a range of 200 of the available 256 bits. This will give us close to 4 V out of the full (theoretical) 5-V range. The core waveform is shown in the bottom left of Figure 1: it's 64 values in '*x*' with a digital range of 0 to 100, or around 2 V.
At this stage we need to use a spreadsheet to calculate the numerical output values for the core data.
The formula for a sinewave is $y = \sin(x)$, where x varies from 0 to 360 degrees or 0 to 2π radians. Unable to figure out how to get Excel to work in degrees here at Elektor labs we just used radians. We've only shown the first three and last three of the 64 core values. To get the values you need to send to the DAC, just multiply $\sin(x)$ by the core range of 100.
Okay, so that's the theory over.

In practice

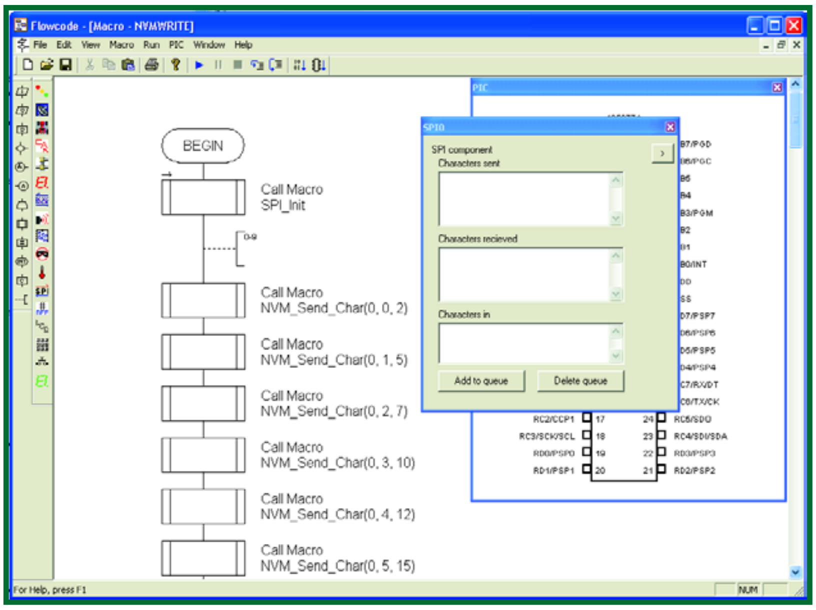
First, let's look at the hardware. We're going to be using the **E-blocks SPI bus Board** shown in the introductory photograph. This goes onto the Multiprogrammer board which happens to have a PIC16F877 on it. You could use almost any PIC with a USART. The SPI board goes on port C and we have an LC display on port B and a keypad on port D.
The SPI board has a MAX5385 DAC as well as a 64-kByte SPI bus and non-volatile memory (NVM). You may first want to get the data into the NVM. As a future extension of the project, consider building up a portfolio of different waveforms inside the NVM and then call them up separately as needed using the keypad and LCD for selection.
To populate the NVM, a program was written that stuffed the core 64 data values into the NVM. A counter from 0–63 was set up, followed by a routine that writes individual bytes of data to sequential NVM locations. In Flowcode this is quite easy, you just initialise the SPI bus and then write a program with 64 icons, each of which writes a byte of data, as shown in **Figure 2**.
The next step will be to write a separate program to check that the core values produce the first quarter of the sine wave. This was fairly straightforward – again a counter is established, reading the value from NVM locations 0 to 63 in turn, followed by a routine forwarding the data from the NVM to the DAC. The Flowcode that did it may be seen in action in **Figure 3**.
The oscillogram in **Figure 4** shows you the first output waveform for the first quarter cycle. It's a bit rough around the edges but looks like a quarter of a sinewave. Unfortunately the time period for a quarter cycle was 3 ms – giving a final frequency of under


Flowcode - [Macro - NVMWRITE]

```
BEGIN
Call Macro SPI_Init
Call Macro NVM_Send_Char(0, 0, 2)
Call Macro NVM_Send_Char(0, 1, 5)
Call Macro NVM_Send_Char(0, 2, 7)
Call Macro NVM_Send_Char(0, 3, 10)
Call Macro NVM_Send_Char(0, 4, 12)
Call Macro NVM_Send_Char(0, 5, 15)
```

Figure 2. NVM stuffing program.

Figure 3. Simple program to take NVM data and send it to the DAC.


Flowcode - [Macro - NVMWRITE]

```
BEGIN
Call Macro SPI_Init
Call Macro NVM_Send_Char(0, 0, 2)
Call Macro NVM_Send_Char(0, 1, 5)
Call Macro NVM_Send_Char(0, 2, 7)
Call Macro NVM_Send_Char(0, 3, 10)
Call Macro NVM_Send_Char(0, 4, 12)
Call Macro NVM_Send_Char(0, 5, 15)
```

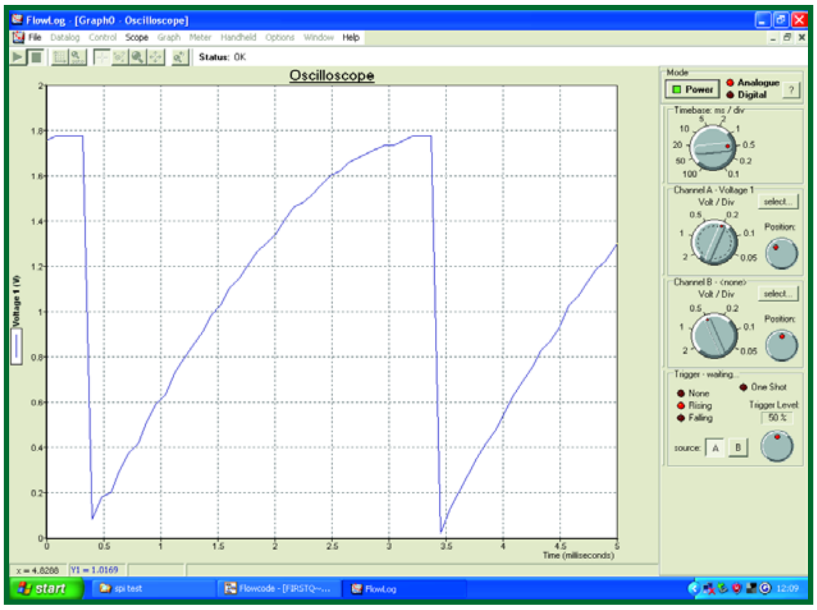
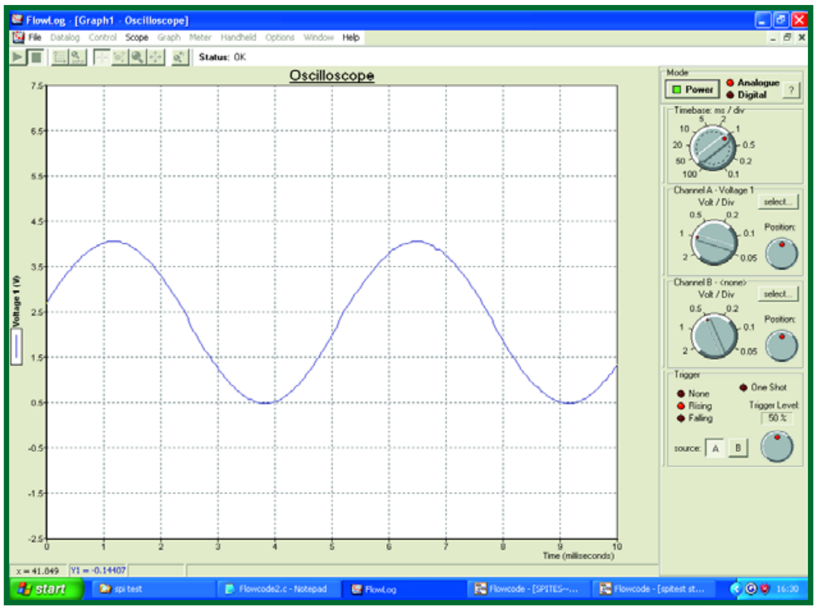


Figure 4. First output waveform showing repeated core values being output.

Table 1 – output values

Sample no.	x	sin(x)	100sin(x)
0	0.024544	0.024541	2
1	0.049087	0.049068	5
2	0.073631	0.073565	7
…	…	…	…
61	1.521711	0.998796	100
62	1.546255	0.999699	100
63	1.570798	1	100

Figure 5.The final output.



100 Hz; too slow for our purposes. Oh well – we have a backup plan.

Towards the goal

Things are slowed down by the need to serially read the data from the external NVM. However, the '877 device we're using also has 256 internal bytes of EEPROM! What we can easily do is fill the PIC's internal EEPROM with our data, which hopefully will speed up the process a great deal.

A new user macro called LOADEEPROM was added to Flowcode. It reads the external NVM and puts the 64 data bytes into internal EEPROM. Consequently the program was altered to read data forwards for the first quarter, and backwards for the second quarter of the waveform. Next, an offset of 127 is added and the data for the second half of the cycle was manipulated similarly. We then measured the results, which you can see in **Figure 5**.

The final signal has a period of 2.5 ms or a frequency of 400 Hz, and a peak-to-peak voltage of around 3.5 V. The waveform does not look too distorted although at the zero-crossing points you can see a small discontinuity. This is presumably caused by program delays between each quarter cycle of data being read out. This may be solved by loading the internal EEPROM with the full 256 databytes and start the waveform at 90 degrees where distortion will have less effect. Let us and others know how you tweaked the project to perfection — post your findings with E-blocks in Elektor's online Forum.

(065031-1)

Earlier in this series

Electronic Building Blocks, November 2005.
E-blocks and Flowcode, December 2005.
E-blocks in Cyberspace, January 2006.
E-blocks — now you CAN, February 2006.

Articles may be downloaded individually from our website.

Programs available for download

SPI memory stuffing.fcf

First quarter.fcf

Sine wave gen.fcf

SPIscreen.FCF

File number: 065031-11.zip

Location: MAGAZINE → March 2006 → 'E-blocks Making Waves'.

In all mains-operated equipment certain important safety requirements must be met. The relevant standard for most sound equipment is Safety of Information Technology Equipment, including Electrical Business Equipment (European Harmonized British Standard BS EN 60950:1992). Electrical safety under this standard relates to protection from

- a hazardous voltage, that is, a voltage greater than 42.4 V peak or 60 V d.c.;
- a hazardous energy level, which is defined as a stored energy level of 20 Joules or more or an available continuous power level of 240 VA or more at a potential of 2 V or more;
- a single insulation fault which would cause a conductive part to become hazardous;
- the source of a hazardous voltage or energy level from primary power;
- secondary power (derived from internal circuitry which is supplied and isolated from any power source, including d.c.)

Protection against electric shock is achieved by two classes of equipment.

Class I equipment uses basic insulation ; its conductive parts, which may become hazardous if this insulation fails, must be connected to the supply protective earth.

Class II equipment uses double or reinforced insulation for use where there is no provision for supply protective earth (rare in electronics – mainly applicable to power tools).

The use of a a Class II insulated transformer is preferred, but note that when this is fitted in a Class I equipment, this does not, by itself, confer Class II status on the equipment.

Electrically conductive enclosures that are used to isolate and protect a hazardous supply voltage or energy level from user access must be protectively earthed regardless of whether the mains transformer is Class I or Class II.

Always keep the distance between mains-carrying parts and other parts as large as possible, but never less than required.

If at all possible, use an approved mains entry with integrated fuse holder and on/off switch. If this is not available, use a strain relief (Figure, note 2) on the mains cable at the point of entry. In this case, the mains fuse should be placed after the double-pole on/off switch unless it is a Touchproof® type or similar. Close to each and every fuse must be affixed a label stating the fuse rating and type.

The separate on/off switch (Figure, note 4), which is really a 'disconnect device', should be an approved double-pole type (to switch the phase and neutral conductors of a single-phase mains supply). In case of a three-phase supply, all phases and neutral (where used) must be switched simultaneously. A pluggable mains cable may be considered as a disconnect device. In an approved switch, the contact gap in the off position is not smaller than 3 mm.

The on/off switch must be fitted by as short a cable as possible to the mains entry point. All components in the primary transformer circuit, including a separate mains fuse and separate mains filtering components, must be placed in the switched section of the primary circuit. Placing them before the on/off switch will leave them at a hazardous voltage level when the equipment is switched off.

If the equipment uses an open-construction power supply which is not separately protected by an earthed metal screen or insulated enclosure or otherwise guarded, all the conductive parts of the enclosure must be protectively earthed using green/yellow wire (green with a narrow yellow stripe – do not use yellow wire with a green stripe). The earth wire must not be daisy-chained from one part of the enclosure to another. Each conductive part must be protectively earthed by direct and separate wiring to the primary earth point which should be as close as possible to the mains connector or mains cable entry. This ensures that removal of the protective earth from a conductive part does not also remove the protective earth from other conductive parts.

Pay particular attention to the metal spindles of switches and potentiometers: if touchable, these must be protectively earthed. Note, however, that such components fitted with metal spindles and/or levers constructed to the relevant British Standard fully meet all insulation requirements.

The temperature of touchable parts must not be so high as to cause injury or to create a fire risk.

Most risks can be eliminated by the use of correct fuses, a sufficiently firm construction, correct choice and use of insulating materials and adequate cooling through heat sinks and by extractor fans.

The equipment must be sturdy: repeatedly dropping it on to a hard surface from a height of 50 mm must not cause damage. Greater impacts must not loosen the mains transformer, electrolytic capacitors and other important components.

Do not use dubious or flammable materials that emit poisonous gases.

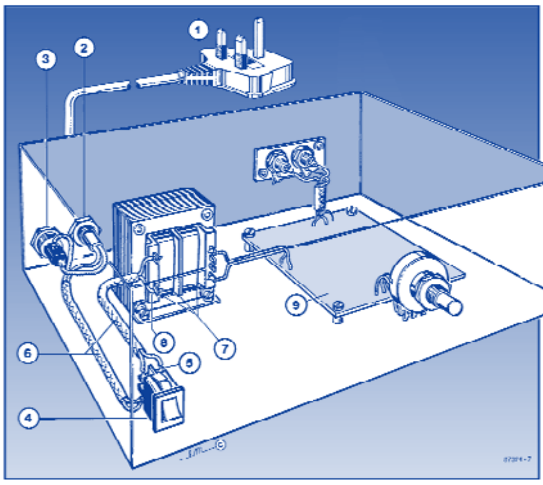
Shorten screws that come too close to other components.

Keep mains-carrying parts and wires well away from ventilation holes, so that an intruding screwdriver or inward falling metal object cannot touch such parts.

As soon as you open an equipment, there are many potential dangers. Most of these can be eliminated by disconnecting the equipment from the mains before the unit is opened. But, since testing requires that it is plugged in again, it is good practice (and safe) to fit a residual current device (RCD)*, rated at not more than 30 mA to the mains system (sometimes it is possible to fit this inside the mains outlet box or multiple socket).

* Sometimes called residual current breaker – RCB – or residual circuit current breaker –RCCB.

These guidelines have been drawn up with great care by the editorial staff of this magazine. However, the publishers do not assume, and hereby disclaim, any liability for any loss or damage, direct or consequential, caused by errors or omissions in these guidelines, whether such errors or omissions result from negligence, accident or any other cause.



1. *Use a mains cable with moulded-on plug.*
2. *Use a strain relief on the mains cable.*
3. *Affix a label at the outside of the enclosure near the mains entry stating the equipment type, the mains voltage or voltage range, the frequency or frequency range, and the current drain or curent drain range.*
4. *Use an approved double-pole on/off switch, which is effectively the 'disconnect device'.*
5. *Push wires through eyelets before soldering them in place.*
6. *Use insulating sleeves for extra protection.*
7. *The distance between transformer terminals and core and other parts must be $\geq$ 6 mm.*
8. *Use the correct type, size and current-carrying capacity of cables and wires – see shaded table below.*
9. *A printed-circuit board like all other parts should be well secured. All joints and connections should be well made and soldered neatly so that they are mechanically and electrically sound. Never solder mains-carrying wires directly to the board: use solder tags. The use of crimp-on tags is also good practice.*
10. *Even when a Class II transformer is used, it remains the on/off switch whose function it is to isolate a hazardous voltage (i.e., mains input) from the primary circuit in the equipment. The primary-to-secondary isolation of the transformer does not and can not perform this function.*

3-core mains cable to BS6500 1990 with three stranded conductors in thick PVC sheath

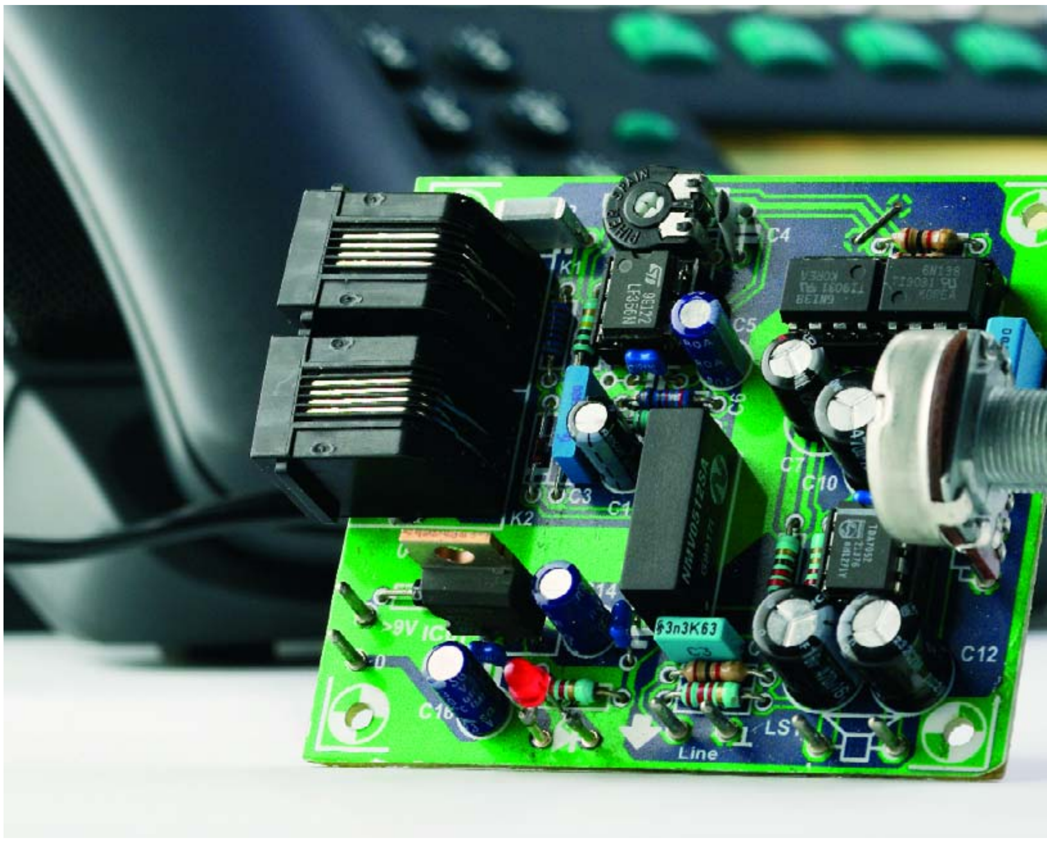
Max current	**3 A**	**6 A**	**13 A**
conductor size	**16/0.2 mm**	**24/0.2 mm**	**40/0.2 mm**
Nom cond area	**0.5 mm^2**	**0.75 mm^2**	**1.25 mm^2**
overall cable dia.	**5.6 mm**	**6.9 mm**	**7.5 mm**

Insulated hook-up wire to DEF61-12

Max current	**1.4 A**	**3 A**	**6 A**
Max working voltage	**1000 V rms**	**1000 V rms**	**1000 V rms**
PVC sheath thickness	**0.3 mm**	**0.3 mm**	**0.45 mm**
conductor size	**7/0.2 mm**	**16/0.2 mm**	**24/0.2 mm**
Nom cond area	**0.22 mm^2**	**0.5 mm^2**	**0.95 mm^2**
overall wire dia	**1.2 mm**	**1.6 mm**	**2.05 mm**

3-flat-pin mains plug to BS 1363A

Telephone Eavesdro



Galvanic isolation is a special feature of this telephone amplifier, which includes a line output connection for a hi-fi system.

From an idea by
Markus Müller

Although in some countries, including the UK and Germany, analogue telephone connections are something of a dying breed, most residential customers use analogue apparatus, even in cases where the underlying connection is over ISDN.

There is little reason to change: first, ISDN telephones are rather expensive, and second, ISDN connection starter packs generally include an a/b adaptor which allows the connection of old analogue telephones to the ISDN system. What a lot of analogue telephones lack, and which most modern ISDN devices provide, is a loudspeaker.

The experienced *Elektor Electronics* reader will have no difficulty in adding a loudspeaker. Several designs for telephone amplifiers have appeared previously in *Elektor Electronics*, but there has never been a circuit to which we cannot make some improvements or add new and interesting features.

High Voltage

The amplifier described here has the special feature, not shared by any of its predecessors, of a line output. This means that it can be connected to an ordinary hi-fi system. This requires, as a glance at the circuit diagram in **Figure 1** will show, complete galvanic isolation between the telephone network and the mains. Now we shall look at the parts of the circuit in turn.
The amplifier has two RJ45 telephone-style sockets connected in parallel at its input, to simplify adding the circuit into an existing cabling arrangement.
An analogue telephone connection works essentially as shown in **Figure 2**. In the quiescent state the exchange provides a DC voltage of nominally 60 V between the a-wire and the b-wire. The actual voltage level varies significantly with the distance from the exchange and can be as low as 40 V or as high as 80 V. To ring the telephone to signal an incoming call, the exchange superimposes an AC voltage of up to 75 V on the DC level. The peak value of the ringing voltage can thus be over 120 V. Despite the low current available, this is dangerous: not just to humans, but also, in particular, to connected electronic devices. When the receiver is lifted, a current of between 10 mA and 30 mA flows and the DC voltage falls to around 12 V. The analogue voice signal is superimposed on this voltage.
Private telephone exchange systems and a/b adaptors can work at much lower DC voltages (often 24 V) because the distance to the telephone itself is much shorter.

Amplification and filtering

Of course, the telephone amplifier must be designed to work at any of these voltages. It is the noble duty of input capacitor C2 to isolate these DC levels from opamp IC1 (an LF356). A high-voltage type is essential, and the component recommended in the parts list can easily cope with 400 V.
Next we must ensure that the AC component is also reduced to a level

pper an advanced amplifier with line output

safe for the input to IC1: this is achieved using two Schottky diodes, D1 and D2, which limit the incoming voltage to between –0.3 V and 15.3 V. The current which flows is limited to a safe value for the diodes (and for the exchange) by R3. The reduced AC voltage (whether it be a ring signal or a voice signal) is taken to the inverting input of inverting amplifier IC1 via R4. Since the useful signal can vary considerably in level, the gain of the amplifier can be adjusted between 0 and 2 (which is slightly lower than might be expected, on account of C2 and C4) using potentiometer P1. At the same time, a DC voltage of 6.8 V is applied to the non-inverting input of the amplifier by voltage divider R1/R2. This voltage can be measured for test purposes at the output of the opamp. This voltage, which is a little less than half the supply voltage, is used at another point in the circuit where it is not allowed to exceed 7 V. The output swing of the opamp is such that this small asymmetry does not matter.

Since the telephone voice signal is not exactly high-quality audio, we can take some liberties with it in the interests of reducing the effects of interference and noise. The input capacitor, along with R3 and R4, forms a high-pass filter with a cut-off frequency of 100 Hz, while C3 and R3 form a low-pass filter with a corner frequency of about 17 kHz. And last but not least, C4 also attenuates higher frequencies above about 10 kHz (or at least those which the opamp is capable of amplifying).

Galvanic isolation

To protect the output stages and any electronics that may be connected to them, two type 6N138 linear optocouplers are used to provide galvanic isolation up to 2.5 kV. Since the performance of the optocouplers is not very dependable (in that their characteristics vary considerably from one device to the next), we use the rather elegant combination of two 6N138s shown to achieve repeatable opera-

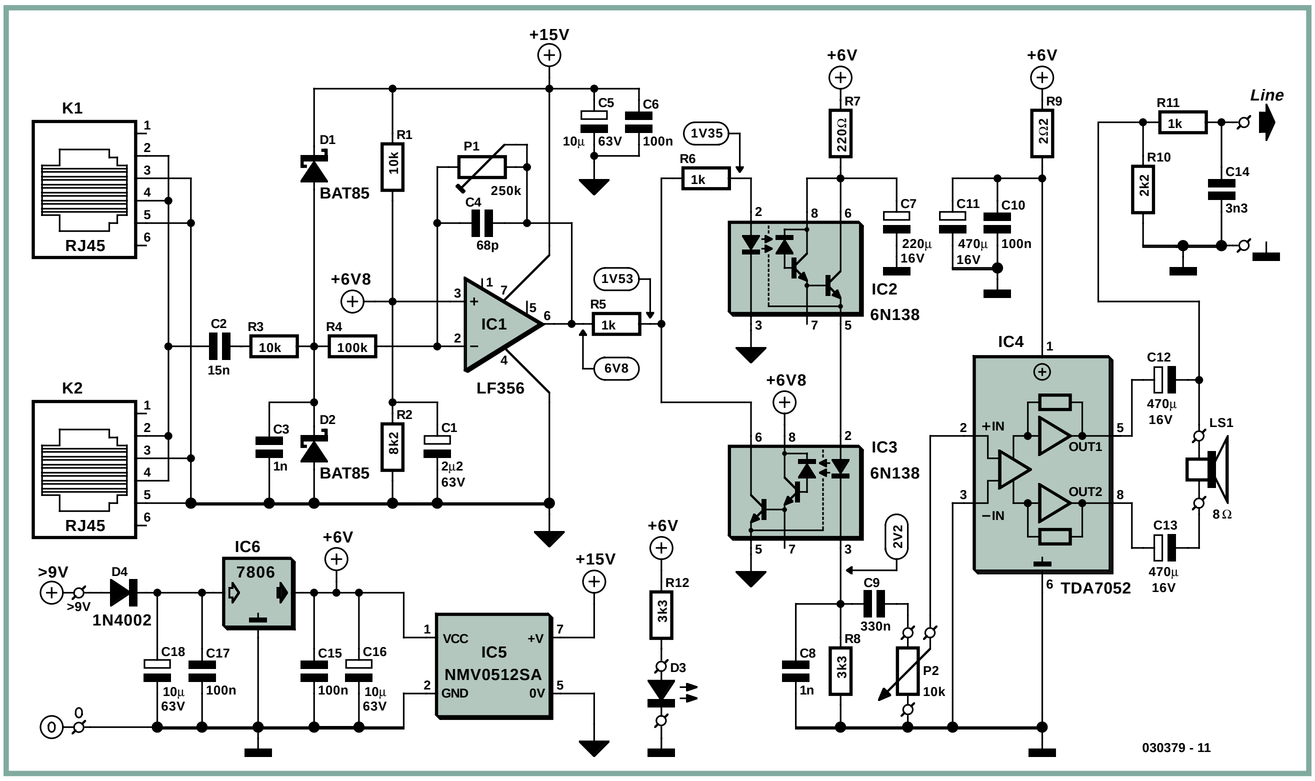


Figure 1. The telephone and audio parts of the circuit are kept strictly separate.

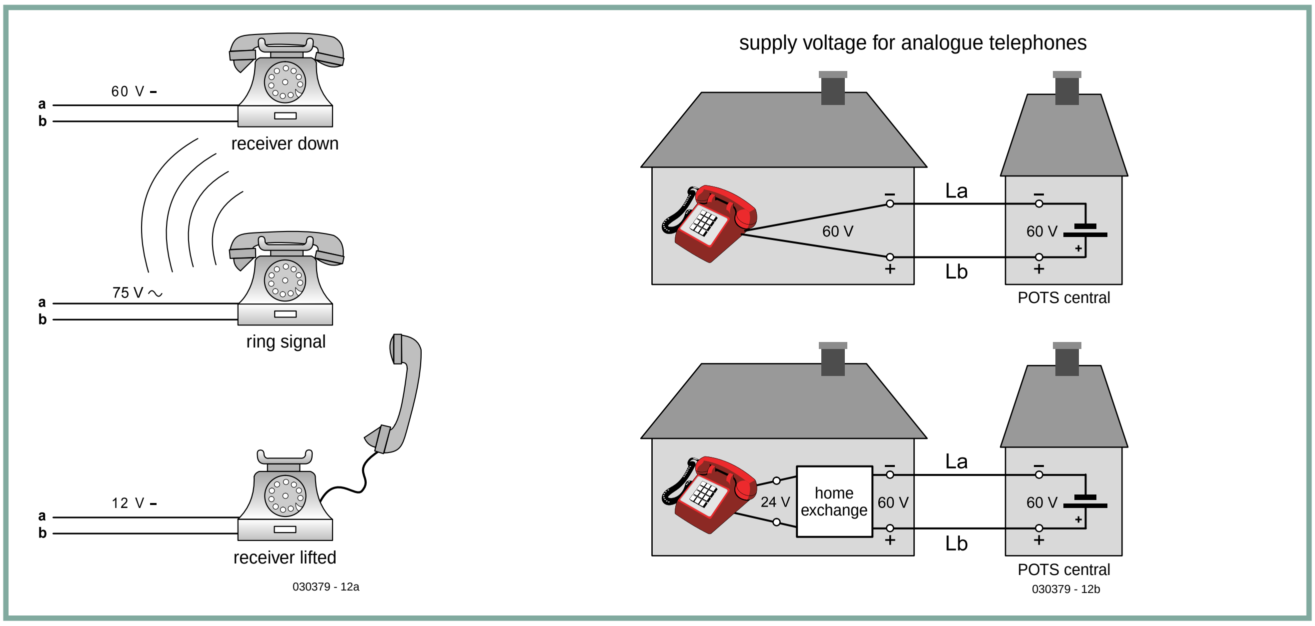
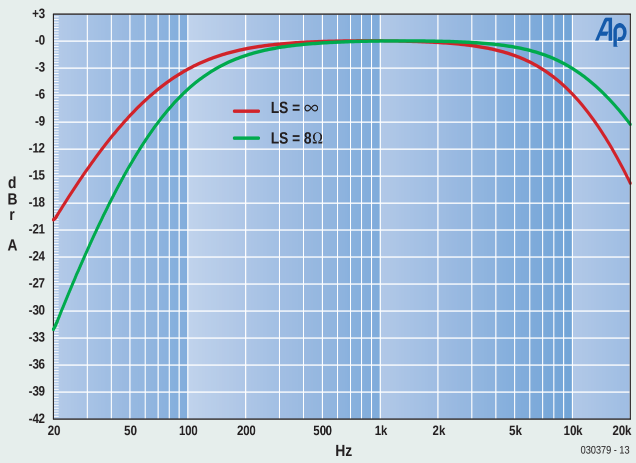


Figure 2. AC and DC voltages on an analogue telephone line.

Measured characteristics

Characteristic curve A shows the frequency response of the telephone amplifier. The measurement was made at the line output, and thus corresponds to the values for the loudspeaker output. The upper frequency range is shown for the two extreme settings of P1, and the effect of capacitor C4 is clearly visible. The –3 dB frequency of 9.9 kHz (at –20 dB) falls to 6.8 kHz when P1 is set to maximum. At the low-frequency end the effect of the loudspeaker is significant. With the loudspeaker connected the corner frequency is 140 Hz, whereas without the loudspeaker it is 100 Hz. Since the output impedance of the TDA7052 amplifier effectively attenuates the output signal by about 1 dB, we have artificially superimposed the curves so that they may be compared more easily.



Here are some of the characteristics as measured on the prototype:

Line output (LS1 not connected)

Sensitivity	Full drive, 1 V line level	60 mV
Maximum output level	THD+N = 1 %	1.6 V
Maximum bandwidth		100 Hz to 9.9 kHz
Minimum bandwidth		140 Hz to 6.8 kHz
Minimum THD+N	B = 22 Hz to 22 kHz	0.26 %
THD+N	B = 22 Hz to 22 kHz, LS1 = 8 Ω	1 %
Signal-to-noise ratio	250 mV in, 1 V out, B = 22 Hz to 22 kHz	55 dB
Signal-to-noise ratio	250 mV in, 1 V out	57 dBA

With LS1 (8 Ω) connected

P_{max} (THD+N = 1 %)	Voice signal	750 mW
Minimum THD+N	B = 22 Hz to 22 kHz	0.6 %
Signal-to-noise ratio	at 500 mW, B = 22 Hz to 22 kHz	55 dB
Signal-to-noise ratio	at 500 mW	57 dB

Quiescent current consumption		72 mA
Current consumption at P_{max}		0.3 A
Current consumption when overdriven	maximum	0.4 A
Minimum supply voltage		9 V

tion. The optocouplers have a Darlington pair at the output and are connected so that the current flowing through the output of IC2 is coupled back via IC3 to the input of IC2. Although this arrangement is slightly noisy, the circuit has adequate bandwidth and the gains in terms of linearity and low distortion more than compensate.

The current though IC2 is translated into a voltage for the following output amplifier by R8. C9 removes the DC component of the signal on R8, and also forms a second high-pass filter with a relatively low corner frequency of approximately 40 Hz. This makes the overall frequency response of the amplifier fairly flat. C8 also reduces interference from the power supply, a point which we shall return to later.
As we stated earlier, the gain of the input stage of the circuit is approximately 2. The attenuation in the optocoupler is approximately 4.7, and so the overall gain up to the volume control potentiometer P2 is around 0.4. This attenuation is not a problem since the gain of IC4 (in bridge mode) is 39 dB. The optimum voltage at the output of the optocouplers, in terms of minimising THD+N, is around 100 mV. At this voltage distortion (the second harmonic is below –53 dB) is comparable with the signal-to-noise ratio of –55 dB. At maximum volume (P2 at maximum), IC4 is overdriven.

Output amplifier

The familiar Philips TDA7052 is used as an output device in this telephone amplifier. This 1-watt bridge-mode amplifier has a low quiescent current and is thus suitable for battery operation. It is powered from the same supply as optocoupler IC2. Bridge-mode amplifiers have the advantage, when used with a single (asymmetrical) supply, that they can be connected to the loudspeaker without coupling capacitors; here, however, we go so far as to fit two such capacitors — and for a good reason!

The electrolytic capacitors form a high-pass filter with a corner frequency of slightly less than 100 Hz, thus removing the lowest frequencies. In theory the corner frequency when using an 8 Ω loudspeaker is 85 Hz, but this value depends, of course, on the impedance of the loudspeaker. In principle a single electrolytic would work

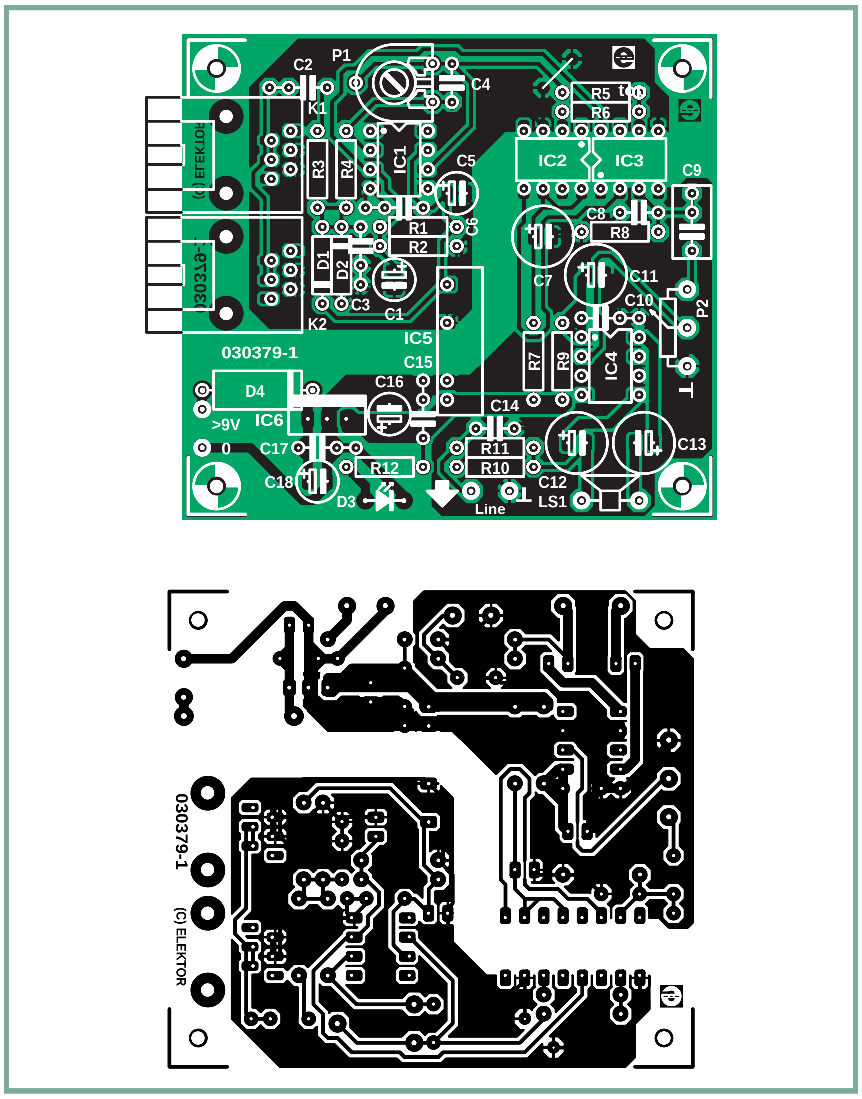


Figure 3. The printed circuit board for the telephone amplifier has two ground planes with a minimum separation of 6 mm between them.

COMPONENTS LIST

Resistors:
R1,R3 = 10kΩ
R2 = 8kΩ2
R4 = 100kΩ
R5,R6,R11 = 1kΩ
R7 = 220Ω
R8,R12 = 3kΩ3
R9 = 2Ω2
R10 = 2kΩ2
P1 = 250kΩ preset H
P2 = 10kΩ logarithmic potentiometer, mono

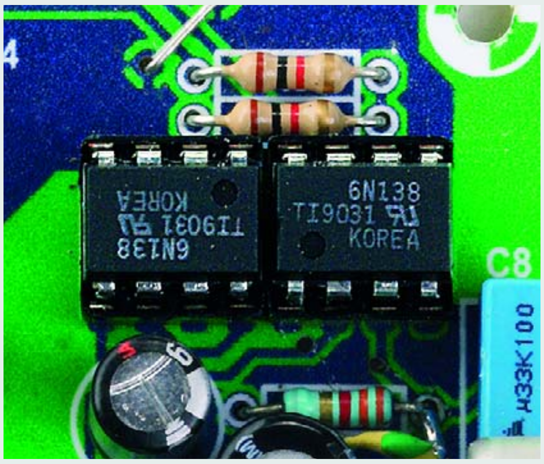
Capacitors:
C1 = 2µF2 63 radial
C2 = 15nF 400V
C3 = 1nF 400V
C4 = 68pF
C5,C16,C18 = 10µF 63V radial
C6,C10,C15,C17 = 100nF
C7 = 220µF 16V radial (max. diameter 8mm)
C8 = 1nF ceramic (5mm lead pitch)
C9 = 330nF
C11,C12,C13 = 470µF 16V radial (max. diameter 8mm)
C14 = 3nF3 (5mm lead pitch)

Semiconductors:
D1,D2 = BAT85
D3 = LED, low current
D4 = 1N4002
IC1 = LF356
IC2,IC3 = 6N138 (Farnell # 325-831)
IC4 = TDA7052
IC5 = NMV0512SA, C&D Technologies/Newport Components (Farnell # 589-822)
IC6 = 7806

Miscellaneous:
K1,K2 = RJ45 socket
LS1 = 8Ω / 1W miniature loudspeaker
Solder pins
PCB, order code **030379-1** (see Readers Services page)

Optocoupler tolerances

Optocouplers have wide tolerances. If possible, use a type 6N138 with a current transfer ratio (CTR) of between 250 % and 450 %, as otherwise IC2 and IC3 may attenuate the signal too much. If the voltage across R8 is a few tenths of a volt too low (a tolerance of 10 % is allowable), its value can be increased to raise the voltage, and vice versa. The circuit is so simple that if necessary other standard optocouplers with a Darlington output, such as the IL300, can be used with a suitable adjustment to the resistor value. The CTR of IC2 is easy to calculate: CTR = (U_{R8}/R8) / (U_{R6}/R6). Changing R8 naturally affects the gain of the circuit, although for reasonable adjustments this should not cause any problems.



just as well, but here the required capacitance is obtained using two capacitors in order to prevent a DC level appearing on the line output, which is taken from one of the amplifier's outputs. Because of this, the gain at the line output is only 33 dB. R10 ensures that the two electrolytics remain charged so that there is no 'click' when an external amplifier is connected to the line output. R11 makes the output short-circuit-proof, and C14 provides additional filtering to remove high-frequency interference. To prevent undesirable coupling through the power supply connections, the optocouplers and the amplifier are separately smoothed (by R7 and C7, and R9, C10 and C11 respectively).

Power supply

There is little point in galvanically isolating the telephone line from the audio electronics if their power supplies are not also kept separate. To this end we use a NMV05125SA DC-DC converter which converts an input voltage of 6 V to an output voltage of 15 V while providing galvanic isolation up to 3 kV.
The input voltage is a little higher than the maximum value of 5.5 V specified in the data sheet, but is clearly within the 'absolute maximum rating' of 7 V. In practice, it works well, and thanks to the low load, the output voltage is slightly higher than the 14.4 V that one would expect with an input voltage of 6 V. The converter presents a rather aggressive load to the fixed voltage regulator, resulting in an output ripple voltage of 0.1 V_{pp}. According to the manufacturer's data, the converter switches at a frequency of approximately 120 kHz. However, in our laboratory prototype the frequency appeared to be a little higher: we measured the ripple frequency to be approximately 300 kHz. This does not affect the performance of the telephone amplifier, but it does simplify the measures we need to take in the circuit to suppress interference.

The fixed voltage regulator is the common-or-garden 7806, with its input protected from reverse polarity connection by a diode. The regulator has a drop-out voltage of 3 V, and so the input voltage obtained from a mains adaptor must be at least 9 V. Of course, the telephone amplifier can also be powered from a battery, although the current consumption is quite high. We measured a quiescent current consumption of 72 mA on the prototype (even using a low current LED for power indicator D3), with peak current up to 0.4 A (using an 8 Ω loudspeaker); battery operation is thus only appropriate if the device is to be used occasionally. In any case, a PP3-style battery will not last long, and it would be better to use four AA cells with IC6 replaced by a wire link and D4 ideally replaced by a Schottky diode such as the 1N5822 (40 V, 3 A). At 0.4 A this Schottky diode drops only 0.3 V, considerably less than the 0.8 V drop of a 1N4002.

Keep your distance

The steps we have taken to achieve galvanic isolation are clear in the printed circuit board layout of **Figure 3**. As can be seen, the ground planes for the telephone and audio parts of the circuit are separated. Populating the board should present no difficulties. Do not forget the wire links (the circuit will work without them, but will be more susceptible to interference). Sockets should be used for the amplifiers but not for the optocouplers, so that the isolation voltage is not compromised.

The connection points for the external components (potentiometer, line socket, loudspeaker and power supply socket) lie around the edge of the board and should be fitted with terminal pins.

The circuit can be built into an ordinary plastic enclosure, but a metallic screen (a piece of tin or unetched printed circuit board) should be fitted directly under the printed circuit board, taking care not to cause a short circuit. The screen should be connected to the metal enclosure of potentiometer P2 and to ground. P2 should be fitted as close as possible to its connection points on the board, and its connection wires twisted.

For best audio results a small, good-quality loudspeaker should be used inside a large enclosure rather than in the electronics box, with the loudspeaker being located a few metres away from the telephone to prevent feedback. Alternatively, the line output can be connected to a hi-fi system.

The circuit discussed in this article is not approved for connection to the public switched telephone network (PSTN) in the United Kingdom.

(030379-1)

Linear Motors
and their modern applications

Dr. Thomas Scherer

A linear motor is simply one which produces motion in a straight line rather than in a circle. We give an overview of the technology and its applications, including examples of 'genuine' linear motors and related devices.

Although space does not permit a full introduction to electromagnetic theory in this article, we should mention that there are linear drive systems where electricity is turned into motion in a straight line which we nevertheless do not count as genuine 'linear motors'. These are so-called dynamic actuators which operate on a very simple principle using a magnet and a coil, and which have long been used in various applications.

A good example of this type of drive is the classical dynamic loudspeaker. **Figure 1** is a diagram of its construction. Its operation is simple and well known: a coil in a magnetic field experiences a force proportional to the current flowing in it. Apart from applications in sound production, this principle has been in use for almost 20 years in the positioning mechanisms for read/write heads in hard disk drives. Once capacities exceeded 40 Mbyte and track widths shrank, a more precise head drive mechanism was required than the rotary stepper motor design like that used in floppy disk drives. Under control of specially-designed position sensors, the 'voice coil' mechanism, as it is known, achieves previously unattainable tracking precision and, thanks to the high force available, fast seek times. In machine tools actuators are starting to be used for simple and short linear motions, replacing pneumatic systems. In these, the roles of coil and magnet are reversed: the coil is fixed and the magnet or piece of metal moves. In contrast to pseudo-linear actuators, where the linear motion is derived from the rotation of a conventional electric motor using a rack and pinion or a belt drive, these solutions have the advantage of being smaller and simpler.

Further advantages include less wear and tear, high power and consequent ease of use and reliability. Since pseudo-linear motors are cheap to produce and can act over greater distances, they will still have their applications in machine tools.

Genuine linear motors

A genuine linear motor differs from its rotary cousins simply in that the stator (the outer cage within which the rotor turns) is flattened out. The rotor then becomes an object which moves along the stator rails or which encloses the one-dimensional stator. Since the translatory motion in a particular direction is produced without the use of rack and pinion mechanisms or the like, this kind of drive is often simply referred to as 'direct drive'.
Therein lies the chief advantage: with many fewer moving parts there is much less wear and tear and practically no play in the mechanism. This all leads to very high reliability.
It is interesting to note that the principle has been known for over 150 years, although industrial use has only been widespread for the last 30 years or so. This is because the great position sensing precision required for machine tools, in the micrometre range, has only recently become achievable using specially-designed semiconductor sensors and high-speed electronic motor controllers.
The enormous power available (accelerations of 20 *g* and more can realistically be achieved in practice) make linear motors a good choice more generally, where rotational motion is not needed.
In theory all the principles employed in rotary electric motors, including stepper motors and three-phase machines, can be applied to linear motors. And, depending on the application, these motors can be bought off-the-shelf in a range of power output levels, maximum displacements and physical constructions. This includes variants with stationary or moving windings, with permanent magnets or electromagnets, and exotic designs for special applications.

Rail guns and catapults

Few technological advances escape application to destructive ends, and linear electromagnetic drives are no exception. **Figure 2** shows a schematic diagram of a gun-like device which, because of its construction, is known as a 'rail gun'. A conductive projectile slides between two copper rails and experiences acceleration when a current flows. During the second world war Germany tried (without success) to construct weapons using this very straightforward principle. Fortunately such weapons are not widely used even today, because of the extremely high currents (thousands or millions of Amps) involved and the great forces that the components of the weapon must withstand.
Using a electromagnetically-propelled projectile as a large-scale catapult is an area of military research. In principle, a catapult would be an ideal way of launching a high-speed aircraft using a scramjet engine. Scramjet engines only start to operate when a relatively high minimum speed is exceeded, rather more than the speed of sound. Experiments to date using working prototypes use a supplementary rocket engine, or the prototype is piggy-backed on a conventional military aircraft and taken to a great height: it is then allowed to fall until it reaches the required speed for the engine to start. With scramjets capable of a speed of up to Mach 15, it is technically reasonable to use a catapult to accelerate them up to several times the speed of sound and start them. It is at least theoretically possible to achieve this using high-powered linear motors (**Figure 3**). NASA has already produced a number of small-scale working research models, sponsored by the military. The system is still, however, science fiction, since the forces cannot be properly controlled, it is extremely expensive and humans are not capable of surviving the necessary accelerations unharmed if the catapult track is short (less than 10 km in length). Practical applications of linear motors to the transport of human beings are taking a different course.

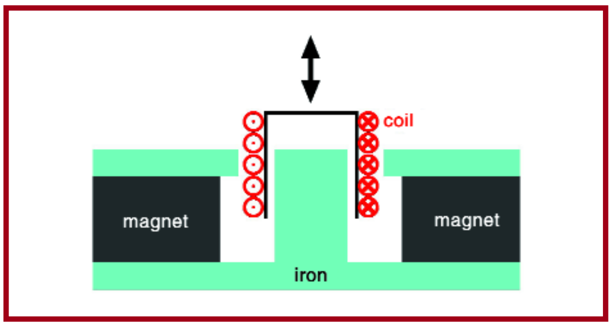


Figure 1.
Principle of the dynamic loudspeaker. This type of drive is used in actuators in industrial applications.

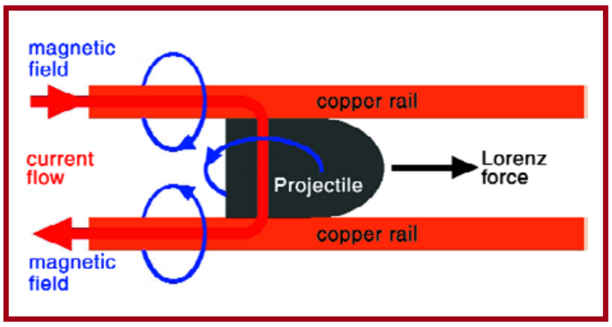


Figure 2.
The 'drive' in a rail gun is very simple. Two current-carrying rails and a conducting projectile are all that is required.

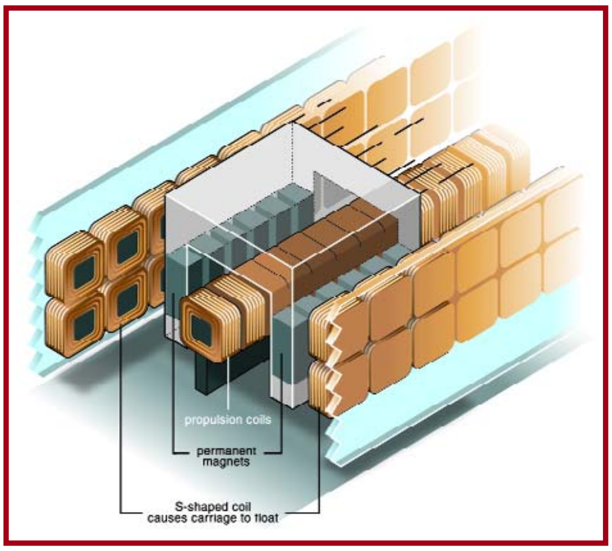


Figure 3.
The linear motor in a catapult. Propulsion and guidance coils allow a metal saddle to be accelerated and keep it on its track. (Source: Schwandt Infographics)

The Transrapid

Relatively high speed trains can be constructed using conventional rotary motors and wheels on rails: the Japanese Shinkansen bullet trains, the German ICE and the French TGV can reach speeds of over 300 km/h. At higher speeds special track must be laid and wear and tear from friction become significant. An experimental version of the TGV in 1990 reached an incredible 515.3 km/h,

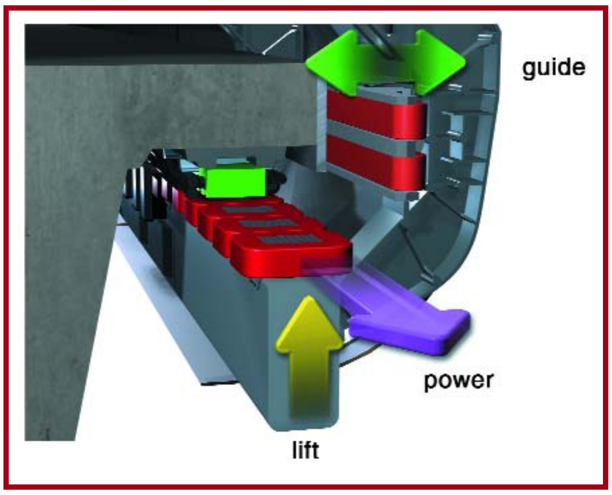


Figure 4. Propulsion and guidance system of the Transrapid train in cut-away view. (Source: Siemens AG)

although at these speeds the overhead wires and pantographs required maintenance after just this one journey. Magnetic levitation railways experience no wear and tear due to friction and it is natural to propel a magnetically-levitated train using a linear motor. Practical versions of this idea are the Japanese JR-Maglev and the German Transrapid. Decades of research have gone into both systems. The Transrapid has been in regular operation in Shanghai for almost two years and has already transported over a million passengers over its 30 km route at some 430 km/h. Both systems are designed for practical use at around 500 km/h and in principle could be run even faster. Consideration is being given to installing a stretch of Transrapid track in the Netherlands (see http://www.magneetzweefbaan.nl/magneetzweefbaan/Default.asp?p=17); in Germany there is still only a prototype track at Emsland in Lower Saxony.

Levitation technology

In the Transrapid design levitation and propulsion are combined. **Figure 4** shows a cross-section of the track and levitation/propulsion system of the train. On the left and right of the track there are metal plates upon which the guidance electromagnets act. The electromagnets responsible for lifting the train act on the laminations which form the flat stator built into the underside of the track on the left and on the right. The air-gap, electronically controlled from within the train itself, is just 10 mm. The windings responsible for propulsion are incorporated into the laminations, made of an isolated aluminium cable with a cross-sectional area of some 300 mm^2. The whole thing functions in the same way as a three-phase synchronous motor laid flat. A travelling magnetic field is induced in the windings in the track. If there is a current in the lifting magnets, they, and hence the train, are lifted up. Since the train envelops the track, it is practically impossible for it to become derailed. **Figure 5** shows a view of the actual construction with the covers removed. Each train or carriage is independently levitated and propelled by the track. There is thus no separate locomotive to pull or push the train. The speed of the train is directly related to the frequency of the alternating currents supplied to the windings, reaching 300 Hz for a speed of approximately 550 km/h in the case of the Transrapid.
It is worth asking how the 10 mm air gap can be maintained at these speeds. The gap is inductively measured thousands of times per second by each segment of the train and the currents in the guidance and levitation electromagnets correspondingly regulated. Taking into account the mass of the carriage a practical control frequency is around 30 Hz. To levitate one segment of the train and guide it around a curve of track requires an electrical power of between 55 kW and 110 kW, depending on load. The necessary energy comes from a suitably-dimensioned battery which is capable of maintaining levitation for approximately one hour if power is lost. If the battery becomes discharged the train lowers itself onto runners. The battery is inductively charged from harmonics of the magnetic propulsion field when the train is travelling at over 100 km/h.
A train composed of three segments requires a drive power of up to 45 MVA at a speed of 500 km/h. So that the whole track is not always powered, it is divided into many short sections selectively driven by inverters placed along the track. The inverters are constructed using power semiconductors such as GTO (gate turn-off) thyristors and IGBTs (insulated gate bipolar transistors) operating at voltages of over 4000 kV and currents of over 1 kA each. The inverter stations control both frequency and voltage and receive commands from a central control station for the track. The Transrapid therefore does not require a driver on the train.
The modern traveller will naturally be concerned as to whether his laptop will operate correctly or whether the data on the magnetic stripe on his credit-card will survive in the neighbourhood of such high-energy magnetic fields. Since the air gap is relatively small at 10 mm the stray magnetic fields are kept to a minimum and the field inside the passenger compartments is comparable in magnitude to the natural magnetic field of the earth. Things are rather different in the case of the Japanese JR-Maglev system where the train rides on a repelling magnetic field. This demands a greater air gap in the region of 10 cm, creating stray fields up to a thousand times greater than those of the Transrapid.
Of course, we have not been able to cover all the details of the systems here. More in-depth information can be found by searching the Internet using keywords such as 'linear motor', 'rail gun', 'scramjet', 'transrapid' and 'maglev'. Wikipedia also has a considerable amount of useful information.

(050374-1)

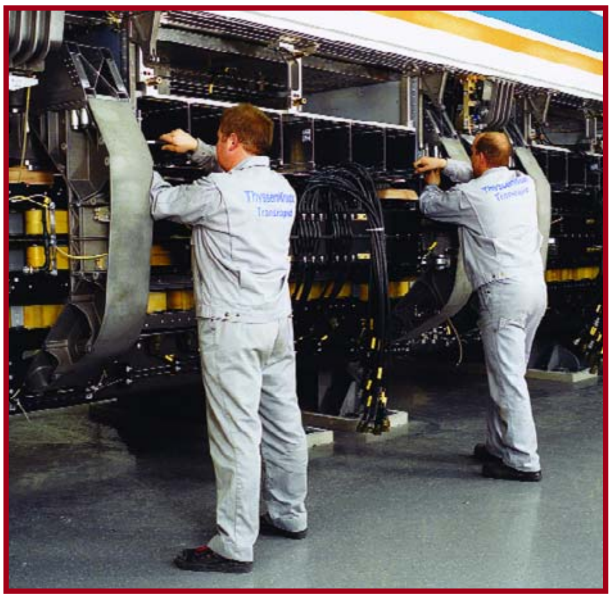
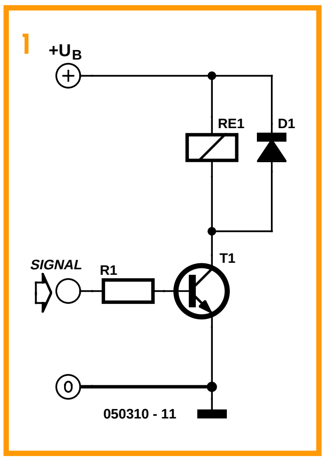
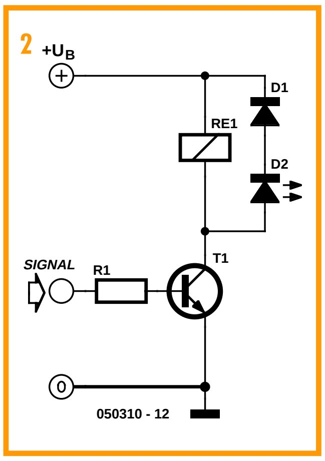
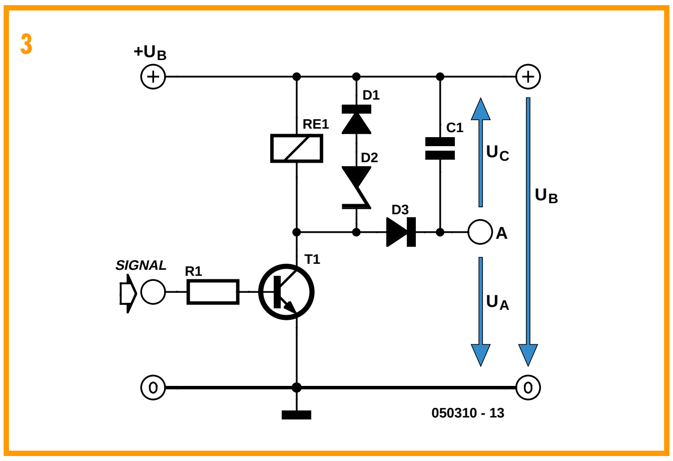


Figure 5. Under the covers of the propulsion and guidance system of the Transrapid train. (Source: Siemens AG)

Energy recovery







Peter Lay

Energy is becoming more and more expensive, and so we are always on the lookout for ways to save energy in circuits. The author has decided to look at how to recovery energy from a relay switching circuit.

If a relay is driven by a transistor switching stage it is usual to connect a flywheel diode in parallel with the coil to short out the back EMF produced when the relay current is switched off (**Figure 1**). If an LED is wired in series with the flywheel diode (**Figure 2**) it will flash every time there is an inductive spike with the transistor turns off. The duration and brightness of the flash (and indeed, whether the spike is large enough to destroy the LED!) depend on the rate of change of the current in the relay coil and its inductance:

$u_i = -L\, di / dt$

So far we have not actually recovered any energy. **Figure 3** shows a theoretical design where the energy stored in the relay coil is recovered so that it can be used to supply a (low-power) circuit. The greater the inductance of the coil and the more frequently it is switched, the more energy is stored in capacitor C. The zener diode (in series with the flywheel diode) limits the maximum voltage to which the capacitor can be charged. Measured relative to ground the open-circuit voltage at point A is the sum of the capacitor voltage due to the recovered energy and the supply voltage. In particular, the voltage at point A is higher than the supply voltage.

The author would be interested in discussing these ideas further with readers. His e-mail address is info@peterlay.de.

(050310-1)

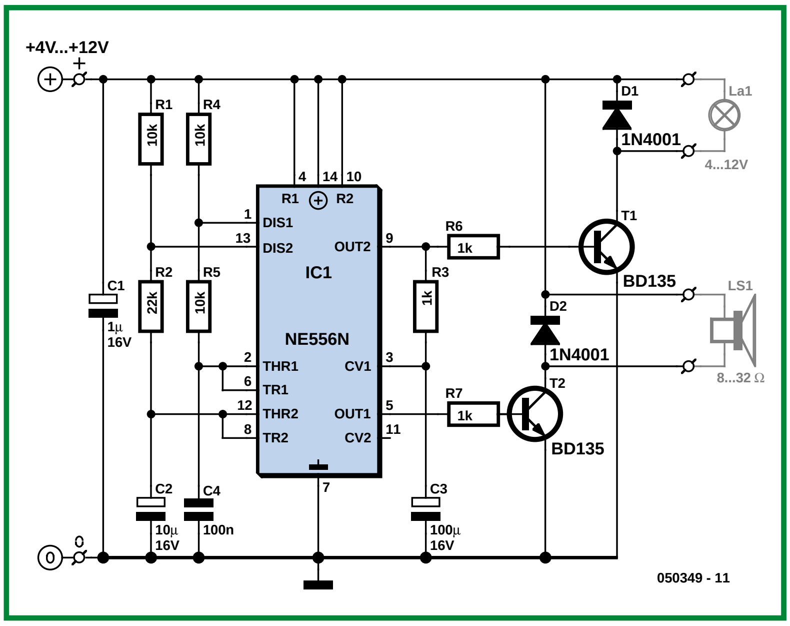
FBI siren with flashing light

Arthur Schilp

This ultra-simple circuit will produce the familiar sound of sirens used by US police cars on emergency calls. A small lamp will also flash synchronously with the siren sound. The circuit is capable of powering loads greater than 1 A for one or more lamps or a powerful loudspeaker, the kit producing quite a bit of noise and light.

The circuit is built from two astable multivibrators, in this case the familiar 555 of which two are present in an NE556 case. Of course, you are free to use two 555s if that suits you better. Both timer ICs are configured to operate as astable multivibrators.

The first timer is configured with R1, R2 and C2 to supply a rectangular signal of about 2 Hz at pin 9. The lamp is switched on and off by way of power transistor T1. The second 555 is configured using R4, R5 and C5, and supplies a square wave at pin 5 that drives the loudspeaker. The toggling voltage at the output of the first timer (pin 9) causes electrolytic capacitor C3 to be partly charged and discharged, periodically, via resistor R3. C3 is connected to the control input of the second timer (pin 3), causing it to work as a VCO (voltage controlled amplifier). The upshot is that the frequency of the square wave applied to the loudspeaker rises and falls periodically, rendering a good imitation of the wailing sound of the US police car siren (we hear too often in movies).



The small number of dead-standard components used enables this circuit to be built on Veroboard without problems.

(050349-1)

Energy saver for relays

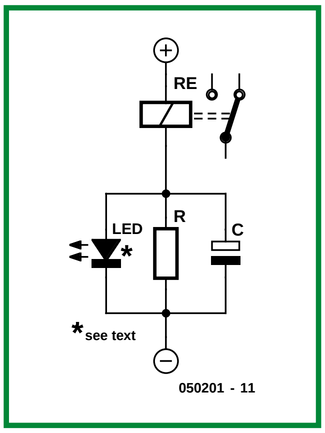
Klemens Viernickel

When dealing with relays we distinguish between the pull-in voltage and the hold voltage. Depending on the type of device, the latter is from about 10 % to 50 % lower than the former. This means that once we have safely pulled in the relay armature we can drop the coil voltage to its hold value, thus reducing the power dissipated. The simple circuit shown here does just that: it consists of a parallel combination of an LED, an electrolytic capacitor and a resistor, together placed in series with the relay coil. As well as saving energy by reducing the operating current of the relay and increasing its operating life the circuit also has the advantage of providing a status indicator in the form of the LED.

The author has used this circuit with practically all types of relay, with various rated currents and voltages. The recommended component values are as follows:

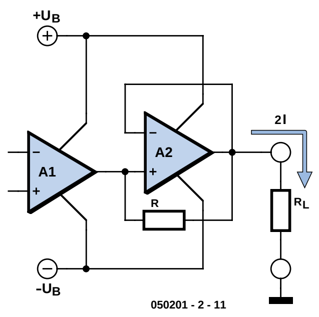
- The electrolytic capacitor should have a value between 100 µF and 1000 µF with a working voltage of 6.3 V, depending on the rated current of the relay coil.
- The resistor value should be between 10 Ω and 1 kΩ so that in the active state a current of 20 mA flows through the LED.
- A standard green or yellow LED with a rated forward current of 20 mA should be used. When using relays with a very low coil current low-current LEDs may be substituted. Add a zener diode in series for higher coil voltages such as 24 V or 48 V.

(050201-1)

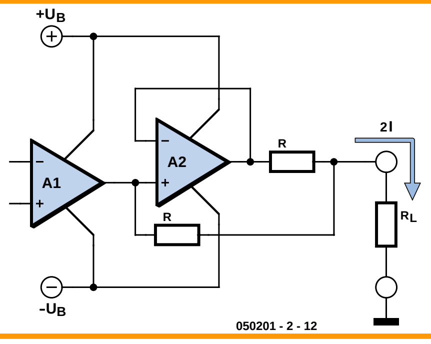


Opamp with increased output current

1



2



Klemens Viernickel

Integrated opamps offer advantages such as ease of use, good price-performance ratio and small physical size. However, they seldom have an output current drive capability of greater than about 12 mA, and so they are not suitable for use in 20 mA current loop applications, for example. One solution is to add a driver stage with the necessary output power, comprising perhaps two to four transistors and a number of other components. This design takes up board space and is relatively expensive, tending to offset the advantages of the integrated device.

An alternative possibility is to boost the output drive capability by connecting opamps in parallel. The output current will then be approximately proportional to the number of opamps. Instead of a single opamp a dual or quad device is used to achieve greater output power.

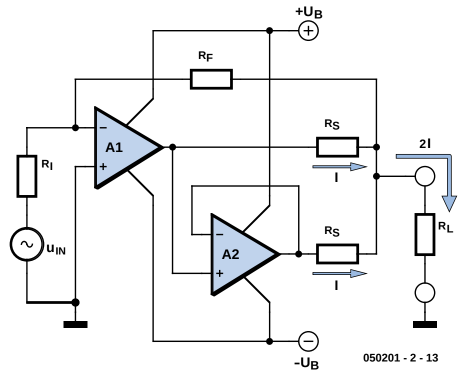
The idea is shown in **Figure 1**. The output of the first opamp is connected to the input of a further non-inverting opamp stage as well as being connected to the output of the circuit via a resistor. The first opamp thus drives the second non-inverting amplifier which provides all the output current of the circuit as long as that remains within its normal capability. As the output current demand increases the second opamp will reach the limit of its drive capability. Its gain will then fall off and a voltage difference will develop across its inputs. The first opamp will then start to deliver more and more current to the output via the resistor, and the sum of the output currents of the two opamps thus flows through load resistor R_L.

By adding another resistor we can compare the current contributions from the two opamps (**Figure 2**). The complete circuit with two opamps is shown in **Figure 3**.

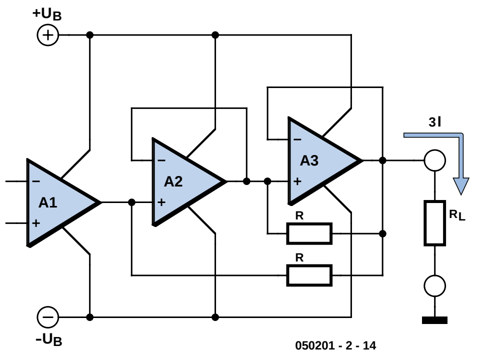
The principle of the circuit can be extended to more opamps with their output currents being added together (see **Figure 4**).

(050201-2)

3



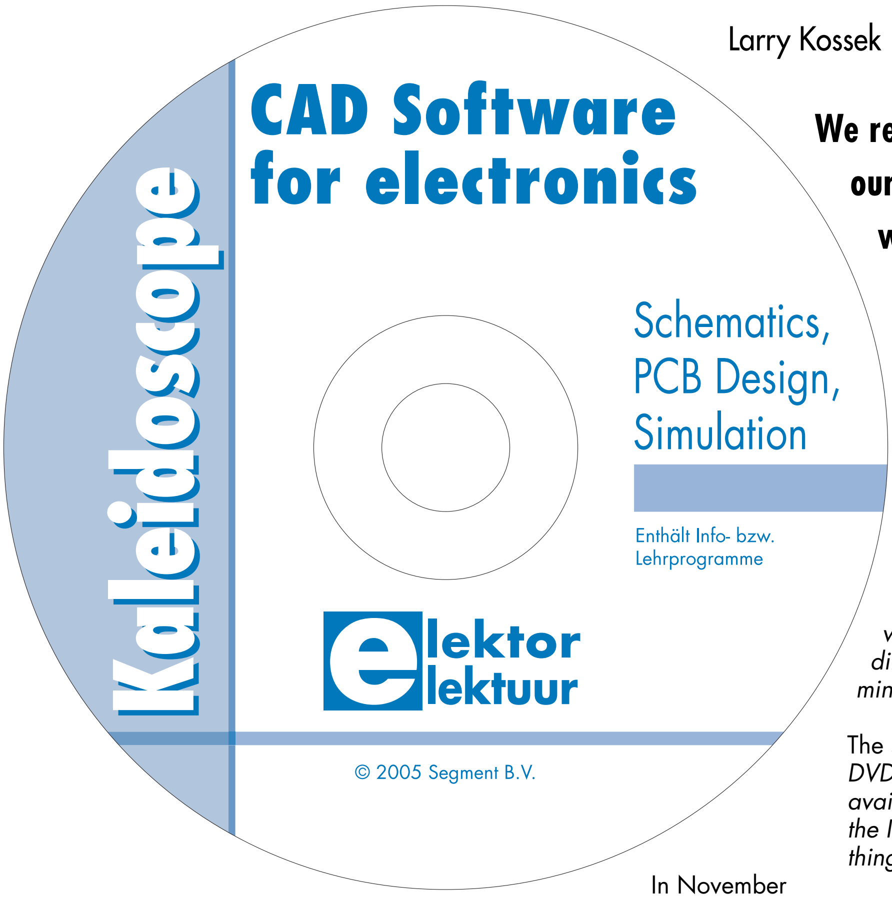
4



The 'Kaleidoscope' E

How was it for you?

Larry Kossek


Kaleidoscope
CAD Software for electronics
Schematics, PCB Design, Simulation
Enthält Info- bzw. Lehrprogramme
elektor elektuur
© 2005 Segment B.V.

We recently featured a readership survey on our website to invite feedback on the DVD which accompanied the November issue of *Elektor Electronics*. This free DVD was a new departure for us and we were very grateful for both the positive and negative comments which landed on our (virtual) doormat...

In November 2005 we included a free DVD with a collection of programs for circuit capture and PCB development/layout along with our magazine. We were keen to gauge your opinion so a readership survey appeared for a few weeks on our website at www.elektor-electronics.co.uk.

On the positive side

Comments ranging from "*Excellent initiative, many thanks*" to "*Thanks very much, I'm a new user, very interesting*" indicated that many of you thought the DVD was a good idea. Appreciative readers also responded: "*This compilation has saved me many hours of work*" or "*Now I have a complete collection of the programs on one disk*"
Although duly announced on our website and in the October 2005 issue, the DVD give-away came as a complete surprise to many readers. The disk was praised for containing "*a good cross section of the tools available*". Several readers commented that they had bought the November issue just for the DVD: "*The DVD alone was worth the cost of the magazine. It has allowed me to make a practical comparison of the available software without additional expense. The relative merits of the different programs can be more easily judged to determine which is more suitable*".

The selection of programs was also well received "*The DVD provided me with a good selection of the software available without the need for lengthy downloads over the Internet (not much fun for a 15 MB program with anything less than broadband)*".

Students on the whole gave an enthusiastic response and even some lecturers commented that "*the DVD contained some tools which we could incorporate on some of our courses at College and University*".
The inclusion of KiCAD was appreciated; it has many followers and can run in either a Windows or Linux environment. Apple users were also surprised to find something for them on the disk: "*Excellent, there's even a Mac program on there!*"

Whether the opportunity to compare many different programs actually led to users switching to a different program package is uncertain. Many comments were along the lines of: "*Interesting and informative but I will continue to use the program that I am familiar with. Using other programs only causes a lot of confusion. I am sure that better software is available but I prefer to stick with what I know*".

One reader mentioned that he had tried out an idea for a front panel design using Frontdesigner from Abacom and liked the result so much that he went ahead and purchased the program immediately after using it for the first time.

-CAD DVD

	DE	NL	UK	FR
Eagle	43%	43%	17%	21%
No Responses	17%	13%	7%	21%
SPlan & SPrintLayout	14%	8%		7%
Target 3001!	9%	3%		5%
OrCAD	4%	8%	11%	11%
Proteus	2%	3%	16%	9%
KiCAD	0%		24%	4%
Layo PCB		7%		4%
Design Suite		4%		

Table 1.

The Eagle has landed

Of all the software on the DVD Eagle was, on average voted the most popular across all four language websites. The UK response shows an anomalous preference for KiCAD but this was probably due to enthusiastic KiCAD fans having teamed up to push the survey results in a certain direction.
The overall results are collected together in **Table 1**. What is not shown on the table is the interest expressed by our readers in so many different programs.

Here we can ascertain that around 60 % of the UK readership indicated that their favourite program was for private use. It was also noted that the software used at home was not always the same software as that used at work. When choosing a program the software costs were considered to be an important factor but not as influential as the opinions of colleagues and friends who were familiar with the software. The simplicity of the user interface is also an important factor; the faster you can begin getting results with a program the more you tend to like it.

The majority of respondents were satisfied or more than satisfied with the programs they were using. Some 10 to 18 % thought that there was probably a better solution that they had not yet found. These readers were particularly grateful for the free DVD in the November issue.

Suggestions and comments

It was clear from the start that we wouldn't be able to provide a copy of every available layout program on our DVD. Many of you suggested **Altium – Protel**, **AutoTRAX EDA**, **DipTrace**, **DouglasCAD**, **Express PCB**, **PADS**, **SupermaxECAD**, **TINA simulator** and **Virtual Breadboard** etc. As reported in Mailbox, a number of CAD software manufacturers simply declined our offer to have a demo or trial version of their product included in the DVD. One reader would have liked a copy of **Scooter** while another would have appreciated comments about the 100-euro version of **BAE**. Circuit simulation software was not included on the DVD but an overview of the topic may be the subject of a future DVD…
Other respondents commented that they were not aware of the existence of some of the programs and were grateful for a chance to try them out.

One parent suggested that it may have been useful to include some simple PCB layout tools that could be used by, say, a 12 year old. This excellent suggestion has

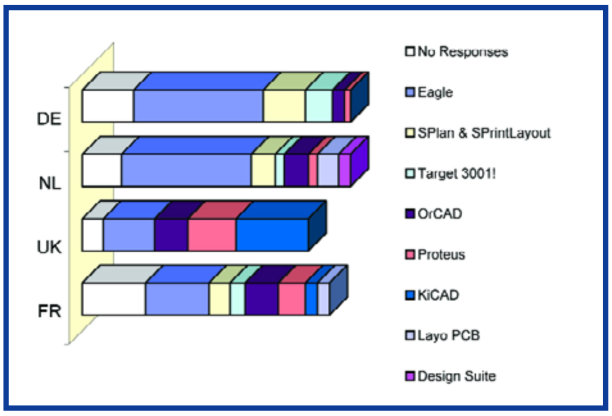
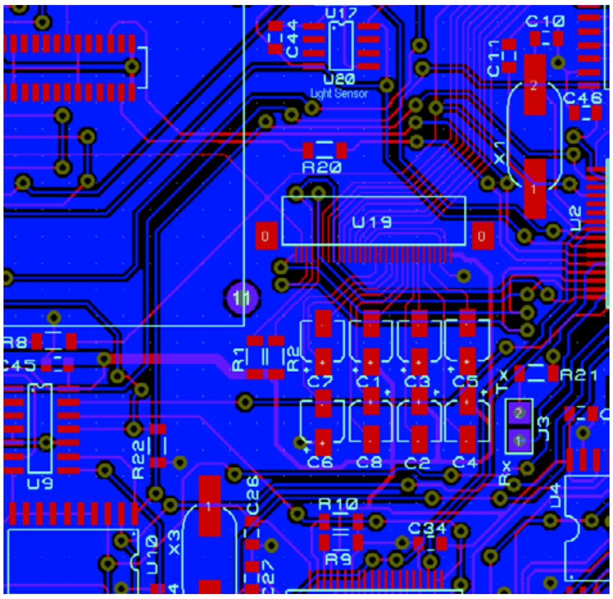


Figure 1.
Program popularity amongst the four European Elektor Electronics websites.



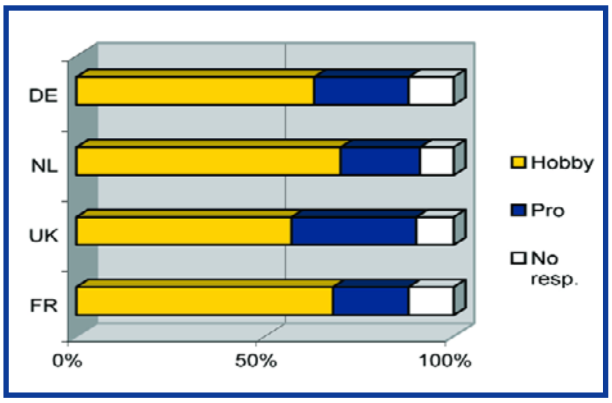


Figure 2.
User profile

been noted and will no doubt be considered when we next get the opportunity to make similar offers.
A KiCAD enthusiast has suggested that we run a tutorial series on this software, good idea, any volunteers out there willing to take this on?

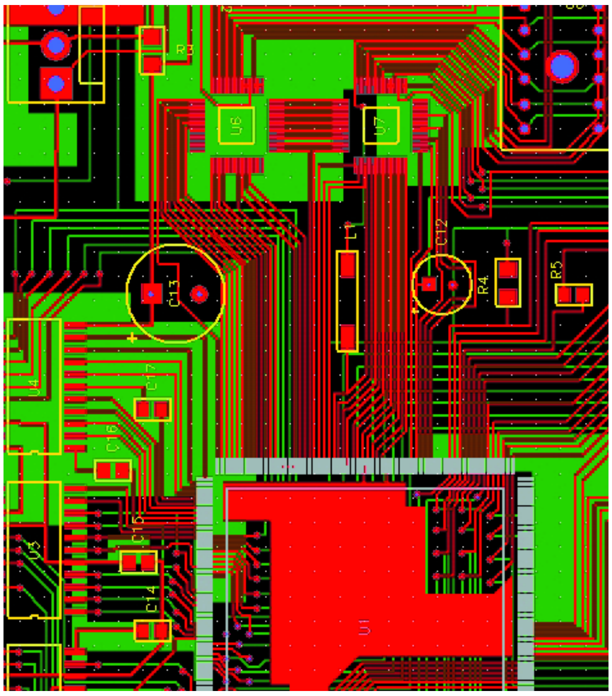
Some programs received critical remarks, the latest offering from Protel for example attracted a number of negative comments while another reader observed that: *"Boardmaker Win is far too expensive for private use!!"*
It came as no surprise that the most popular programs attracted praise (e.g. EASY PC), and especially SPlan and Sprint Layout were often mentioned.
"SprintLayout fulfils my requirements more precisely than any of the other programs".
One reader suggested that a future DVD might feature PC-based test equipment and circuit simulation software. Another reader thought a DVD containing programming tools and program development environments would be useful. These suggestions have been noted and may be incorporated in future DVDs.

Some readers commented that most of the programs are demo versions so it is difficult to make a meaningful comparison between the software especially if you are not aware of the price of the full software package: *"it would not be fair to compare a 200 pound program with a 10,000 pound without taking into account the price of the software."*
We would beg to disagree on this point and suggest that program comparisons are more likely to be 'objective' if the cost of the software is not considered. However we take on board that it may have been useful to include a table indicating price / demo / freeware information.

You can't please everyone...

Negative reactions were also registered, some readers complained that the DVD was missing from their issue and these have hopefully been supplied free of charge from the publishers by now.
Some readers reported that they could not run the DVD. Many of the problems turned out to be "user induced" ranging from surprise when their CD-ROM drive was unable to run the DVD (I must admit I fell into this trap when I tried to run the DVD at home on my ancient laptop) to *"I can't see anything on the TV when I plug the disk into my DVD player"* a mistake which can probably be attributed to the effect on the brain of too much partying in the run up to Christmas.
Disks damaged in transit were quite rare and have been replaced.
Some of the more extreme reactions included comments that the DVD was *"completely unnecessary"* or *"totally useless"*. Other comments judged the disk contents to be *"meagre"*.
One problem that we cannot be held responsible for is a symptom of the well known "Windows disease". It is possibly one of the most frustrating features of some Windows programs that after trialling software and subsequent de-installation, traces of the original program can still be found on the computer and are notoriously difficult to remove. This is however not the case for all programs, some are more well behaved than others.

A number of you would have liked to see a more detailed article accompanying the disk: *"It was disappointing that the comparisons were sketchy and inadequate, but nevertheless it was a splendid effort regardless"* Another commented: *"many of the programs require a large investment of time and the limited nature of the demo software means that it was not worth the time and effort."* Reading between the lines it seemed that many respondents were looking for advice on the best system to purchase, but such recommendations were not what we intended and are anyway almost impossible to make for this type of software.
Many had hoped for fuller versions of the software but we can only pass on the version which is supplied to us. It will only take one publisher to release a freeware version with more features for the others to follow suit to maintain market share.



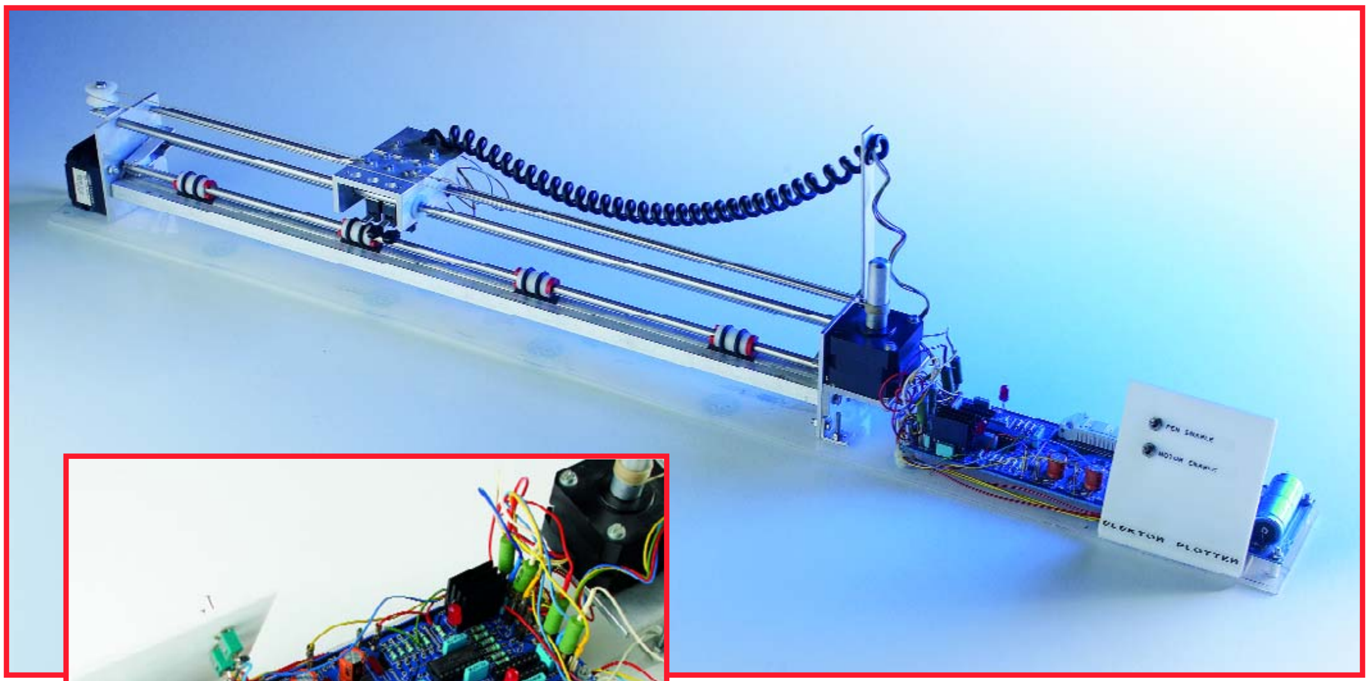
A tale of woe

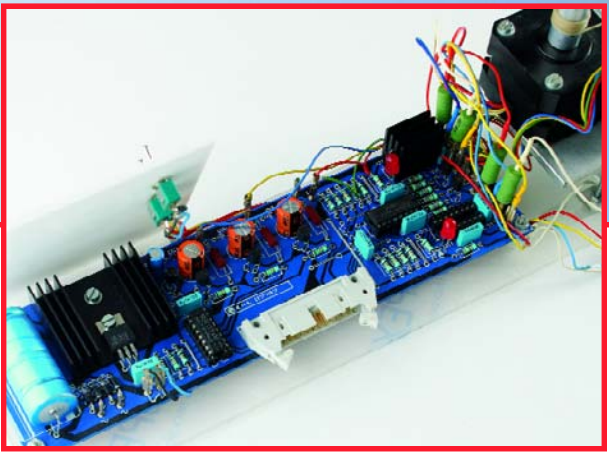
Quote: *"I have been using the program Capture and Layout for a number of years and from day one I have been looking forward to the next software release that would surely improve the program. Unfortunately it has not been such a simple procedure; Features that were good in the previous version of the software have been omitted in the latest version. Known bugs were fixed but it seemed like just as many new bugs were introduced! Multiple 'undo's' are implemented in Capture 10.0 but only work for a few, less important commands. In Layout, like previous versions there is only one or no step forward or back. This problem was identified in a review of the program in Elektor Electronics many years ago. This has been my experience with a CAD program made in the USA."*
Who would have thought that PCB layout could be such a headache?

Taking into account the positive and negative comments it seems that on the whole the disk was received favourably. We have gained much from this experience and once again would like to thank all those who participated.

(050323-1)

Elektor Plotters (1988-1991)





Jan Buiting

This story starts in May 1988 with the publication of an article with the delightfully simple and technically correct title 'Plotter (part 1)". The design was based on a mechanical drawing and an accompanying letter sent to us by one J. Arkema from the Netherlands. This was at a time that plotters were noisy, slow and expensive PC adjuncts only seen in the cellars of professional engineering bureaus running AutoCAD or similar programs. Of course, the electronics hobbyist had different things in mind to do with a plotter, if only he could get his hands on one: drawing schematics and 'printing' PCB artwork! Unfortunately, to this day a working high-end plotter, particularly one of Hewlett Packard, is a fairly rare find.

The proposed plotter, then, was a simple X-Y type, with the Y movement carried out by a platen moving the paper sheet (up to A2 size!), and the X motion handled by a pen carriage covering a span of about 50 cm along a guide bar. The May 1988 article came with a set of mechanical drawings of the (mostly aluminium) parts to cut, mill, drill and file before you could even start dreaming of assembling the plotter. These drawings were produced manually on an ordinary drawing stand. It turned out a quite unusual job for our drawing staff, and it took several versions before a 'blueprint' was produced that could be read by someone accustomed to working with a lathe. Mind you, no plotter was available, let alone a program or a PC.

Unfortunately, despite the efforts that went into producing the drawings, a poor balance was struck between the mechanical construction (wrapped up in about two pages) and the electronics we all understood so much better. Although a mechanical parts list was printed and an artist's impression of the assembled plotter, we were inundated with questions like 'how to assemble the thing' and where to obtain Skiffy parts, Binder magnets, Berger stepper motors and Rotring pen refills. Fortunately, one of our advertisers, Meek-it from The Hague, Holland, stepped in and offered a kit of parts to build the plotter.

After al the hubbub around the mechanical construction, silence reigned when we published part 2 of the article in June 1988. In it, we duly described the algorithms that (we assumed) would enable talented readers to write their own drivers to control the plotter from a PC. A wrong assumption, as virtually no response was received and for quite some time it seemed that no one had the fuzziest notion of what we meant by octants, stepper motor look-up tables and MC3479 control bytes, let alone Bresenham's algorithm.

The silence was broken in March 1990 when we were finally able to publish "Plotter Mark 2", discussing a few mechanical improvement to the original design but more importantly a 'partly HPGL compatible' driver for PCs! The program from B. Lewetz was called *Mondriaan* after the Dutch artist (1877-1944). Although it supported just six basic commands from the much larger HPGL set, the program effectively made the project come alive after a two year wait. Reader response was overwhelming as the missing link between hardware and software had been catered for at long last. The famous 'Columbia' and 'nozzle' drawings were supplied with the program, along with a few pen calibration and plotter test files. Market leading programs like Autocad and Autosketch at the time employed only six HPGL commands so the Elektor plotter did just fine on the Centronics port. In the UK, Cliff Gregory optimised the mechanical design of the Elektor plotter and started to supply 'Plotter Mk3' kits comprising ball bearings and stainless steel parts.

Finally, in September 1991 we published another plotter driver, this time written in Turbo Pascal and supporting 17 HPGL commands. Supplied on a 360 kB 5.25 inch floppy disk (and later on 3.5 inch), it turned out to be huge sales success.

(065015-1)

Retronics is a monthly column covering vintage electronics including legendary Elektor designs. Contributions, suggestions and requests are welcomed; please send an email to editor@elektor-electronics.co.uk, subject: Retronics EE.

Hexadoku

Puzzle with an electronic touch

Here is the third Hexadoku puzzle, the brain teaser for electronics fans and their family members. Solve the puzzle and win one of the fantastic prizes!

The instructions for the puzzle are straightforward. In the diagram composed of 16_16 boxes, enter numbers in such a way that **all** hexadecimal numbers 0 through F (that's 0-9 and A-F) occur once in every row, once in every column, and in every one of the 4_4 boxes (marked by the thicker black lines). A number of clues are given in the puzzle and these determine the start situation.

Your solution may win a prize and requires only the numbers in the grey boxes to be sent to us (see below). The puzzle is also available as a **free download** from our website (Magazine → 2006 → March).

(065043-1)

	D	1	A		C		9	4	2	B	8	F			7
				5	3	8				0	7		6		
			2		4		7		6		C		A		
			0					3				2	4		9
			D			6	▨	▨	▨	▨	▨	0			2
2	8		F	4	9	D					E		7	A	6
	3	7	B	F		A		D		2	6	9		4	1
		4		2		7		8	0	F	A	5		D	
			7	1		4					D	C	B		0
	9	0		B		5					F	1			4
B		2	5		E				1			6	8	9	
C	4				2		8		E	5	B				A
9			4		0	2	1		7				D		
A		F	6		5										8
7				8	6			9		A			0		F
		B	E				A	0	5				9	6	C

Entering the competition

Please send the numbers in the grey boxes by email, fax or post to

Elektor Electronics Hexadoku
Regus Brentford
1000 Great West Road
Brentford TW8 9HH
United Kingdom.
Fax (+44) (0)208 2614447
Email:
editor@elektor-electronics.co.uk
Subject: hexadoku 03-2006.

The closing date is **27 March 2006**.
Competition not open to employees of Segment b.v., its business partners and/or associated publishing houses.

Prize winners

The solution of the January 2006 Hexadoku is: EA639.
The **E-blocks Starter Kit Professional** goes to: Ian Mowatt (Chester).

An Elektor SHOP Voucher worth £35.00 goes to: M. Devereux (Stockton on Tees); Paul Ackerman (Broughton Ashley); Colin Wilkinson (Wareham).

Solve Hexadoku and win!

Correct solutions qualify for an

E-blocks Starter Kit Professional

worth **£248.55**
and three
Elektor Electronics Shop Vouchers
worth **£35** each.

We believe these prizes should encourage all our readers to participate!

Theme Plan for 2006

January	Recycling / Reverse Engineering
February	Motors / Propulsion
March	Development / Microcontrollers
April	**Power Supplies / Safety**
May	Soldering / Etching
June	Satellites
July/August	Summer Circuits
September	Esoterics / Test & Measurement
October	e-Simulation
November	Chipcards / Protection
December	Electromechanical / Enclosures

SPI Box

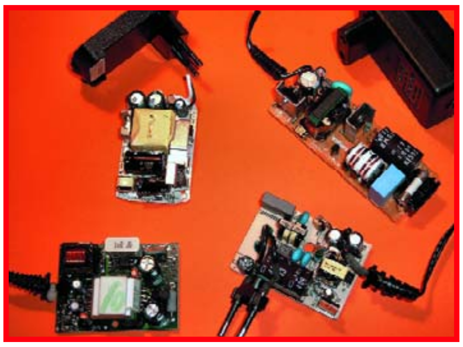
Many of today's microcontrollers and associated peripheral ICs sport a Serial Peripheral Interface (SPI) connection for programming and control purposes. The SPI is a relatively simple add-on and requires little hardware to drive from a PC. Usually, an RS232 port is 'misused' for the purpose but admittedly that's fraught with a few disadvantages. The programmer we've in mind goes round these problems while also ensuring short programming times even if a USB/serial adapter is being used for lack of an RS232 port on the PC or laptop.

The Electronic Laundry

Microcontrollers are applied in massive numbers in all sorts of consumer-market electrical equipment. Large manufacturers spend lots of timer and money on hardware and software development for a specific product. Eventually, units have to be mass-produced by the millions so design errors and bugs are out of the question. We paid a visit to of washing machine giant Miele for a peek in their hard- and software development department.

Mains Adapters Undressed

Switch-mode versions of the trusty mains adapter (or 'battery eliminator') are increasingly seen with mobile equipment like PDAs, phones and MP3 players. The newer versions no longer have bulky, energy wasting mains transformers and linear regulators. We took a few of these newcomers apart to examine their construction and operation. The experience so gained is put into practice by the home construction of a DC-DC converter.



Also…

Electric Fence; AA Cell Characteriser; FPGA Applications (1); E-blocks; Hexadoku Puzzle.

RESERVE YOUR COPY NOW! The April 2006 issue goes on sale on Thursday 16 March 2006 (UK distribution only).
UK subscribers will receive the magazine a few days before this date. Article titles and magazine contents subject to change.

NEWSAGENTS ORDER FORM

SHOP SAVE / HOME DELIVERY

Please save / deliver one copy of *Elektor Electronics* magazine for me each month

Name:	. . . .
Address:	. . . .
Post code:	. . . .
Telephone:	. . . .
Date:	. . . .
Signature:	. . . .

elektor electronics leading the way

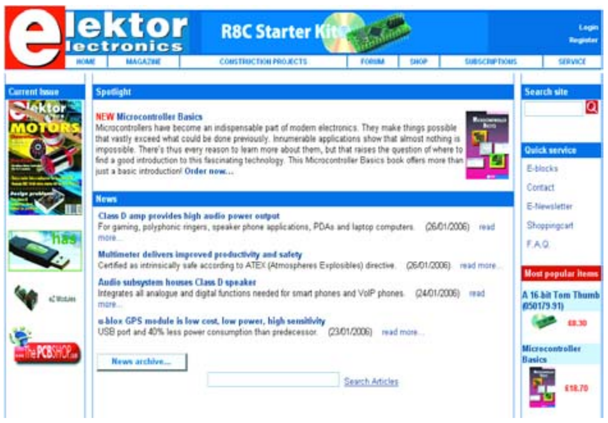
Please cut out or photocopy this form, complete details and hand to your newsagent. *Elektor Electronics* is published on the third Friday of each month, except in July. Distribution S.O.R. by Seymour (NS).

www.elektor-electronics.co.uk www.elektor-electronics.co.uk www.elektor-electronics.

Elektor Electronics on the web

All magazine articles back to volume 2000 are available online in pdf format. The article summary and parts list (if applicable) can be instantly viewed to help you positively identify an article. Article related items are also shown, including software downloads, circuit boards, programmed ICs and corrections and updates if applicable. Complete magazine issues may also be downloaded.

In the Elektor Electronics Shop you'll find all other products sold by the publishers, like CD-ROMs, kits and books. A powerful search function allows you to search for items and references across the entire website.



NEW Microcontroller Basics
Microcontrollers have become an indispensable part of modern electronics. They make things possible that vastly exceed what could be done previously. Innumerable applications show that almost nothing is impossible. There's thus every reason to learn more about them, but that raises the question of where to find a good introduction to this fascinating technology. This Microcontroller Basics book offers more than just a basic introduction! Order now...

News

Class D amp provides high audio power output
For gaming, polyphonic ringers, speaker phone applications, PDAs and laptop computers. (26/01/2006) read more...

Multimeter delivers improved productivity and safety
Certified as intrinsically safe according to ATEX (Atmospheres Explosibles) directive. (26/01/2006) read more...

Audio subsystem houses Class D speaker
Integrates all analogue and digital functions needed for smart phones and VoIP phones. (24/01/2006) read more...

u-blox GPS module is low cost, low power, high sensitivity
USB port and 40% less power consumption than predecessor. (23/01/2006) read more...

A 16-bit Tom Thumb (050179-91) £8.30
Microcontroller Basics £18.70

Also on the Elektor Electronics website:

- Electronics news and Elektor announcements
- Readers Forum,
- PCB, software and e-magazine downloads
- Surveys and polls
- FAQ, Author Guidelines and Contact

C booklet

reinhardt weber

```
setup_r8c()
{
   prc0 = 1;             /* Protect off */
   cm13 = 1;             /* Xin Xout */
   cm15 = 1;             /* XCIN-XCOUT drive cap
   cm05 = 0;             /* Xin on */
   cm16 = 0;             /* Main clock = No divi
   cm17 = 0;
   cm06 = 0;             /* CM16 and CM17 enable
   asm("nop");           /* Waiting for stable
   asm("nop");           /* Assembler code
   asm("nop");
   asm("nop");
   ocd2 = 0;             /* Main clock change */
   prc0 = 0;             /* Protect on */
   pd1 = 0x0F;           /* Set Port 1.0-1.3 be
}


toggle_leds()
 {
while (1)
{
p1   = 0x00;
```

Not just for R8C fans!

elektor electronics
leading the way

Contents

Foreword

Why C?

Many electronics hobbyists have used microcontrollers successfully, and they have also written wonderful programs in assembly language. As the size and complexity of an assembly-language program increase, the desire for a more effective programming environment also increases. Anyone who has tried to implement a mathematical function in assembly language, such as 1/x, sin(x) or the like, knows the problems. Here a high-level language such as C, which is the industry standard in the microcontroller and microprocessor world, offers decisive advantages. C programs are portable, which means the program structure can be transferred to other types of microcontrollers after it has been written. The only things that have to be modified are the port assignments and the settings for the special function registers.
Professional programmers claim that an assembly-language program that would take 14 days to generate can be generated in 2 to 3 days in C. What's more, an increasing number of semiconductor manufacturers are making highly effective development environments available at no charge. That's another good reason to start using C.

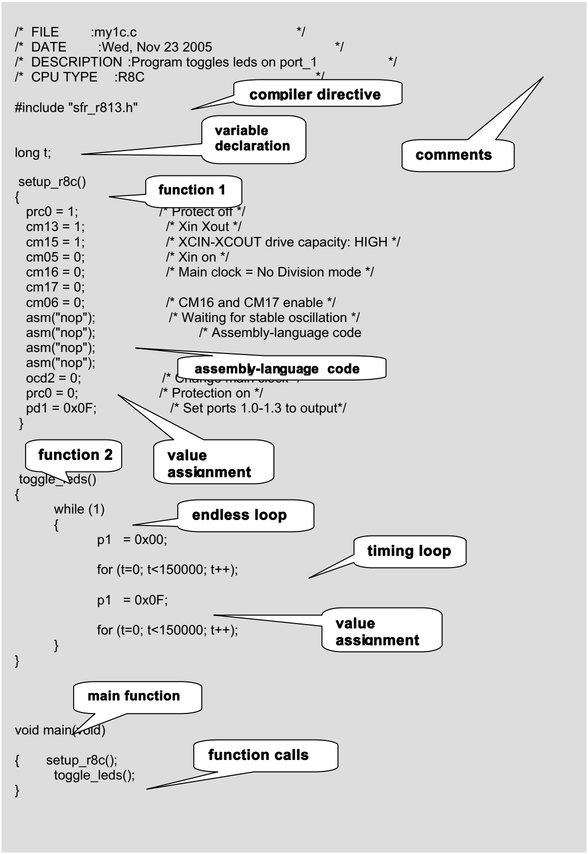
But there's a hitch. As we all know, the gods have ordained that success doesn't come without hard work. An introductory course in the C language and sample programs from technical magazines can help you overcome the initial hurdles, but will take a while before you can write you own programs. You will also have to master a certain amount of specialist vocabulary. As can be seen from contributions to microcontroller forums and questions asked in these forums, that forms a significant problem for many electronics hobbyists.

This booklet is limited to the basic elements of the C language. We have intentionally omitted complex C structures such as pointers, arrays, strings, structures, unions and the like. This booklet is intended to serve as a reference for beginners. It cannot replace a basic course in C, nor is it intended to do so.

C Basics

The structure of a C program

All C programs consist of several parts, such as comments, preprocessor instructions, declarations, definitions, expressions, assignments and functions. The following listing shows a simple example.

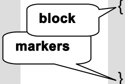


```
/*  FILE        :my1c.c                                    */
/*  DATE        :Wed, Nov 23 2005                          */
/*  DESCRIPTION :Program toggles leds on port_1            */
/*  CPU TYPE    :R8C                                       */

#include "sfr_r813.h"


long t;

 setup_r8c()
{
   prc0 = 1;                /* Protect off */
   cm13 = 1;                  /* Xin Xout */
   cm15 = 1;                  /* XCIN-XCOUT drive capacity: HIGH */
   cm05 = 0;                  /* Xin on */
   cm16 = 0;                  /* Main clock = No Division mode */
   cm17 = 0;
   cm06 = 0;                  /* CM16 and CM17 enable */
   asm("nop");                 /* Waiting for stable oscillation */
   asm("nop");                       /* Assembly-language code
   asm("nop");
   asm("nop");
   ocd2 = 0;                 /* Change main clock */
   prc0 = 0;                /* Protection on */
   pd1 = 0x0F;                /* Set ports 1.0-1.3 to output*/
 }


 toggle_leds()
{
        while (1)
        {
                p1   = 0x00;

                for (t=0; t<150000; t++);

                p1   = 0x0F;

                for (t=0; t<150000; t++);
        }
}



void main(void)

{       setup_r8c();
        toggle_leds();
}
```

The main function

Every C program must include at least one function, which is called the main function. This is the primary function in a C program, and it is always the first function to be called when the program is run. It's considered good programming style to have the main routine consist primarily or entirely of function calls, instead of containing the entire code of the program. That makes the program a lot easier to understand and maintain, and it allows the programming effort to be divided among several programmers for large projects. The main function is declared in the same manner as any other function.

```
void main(void)
{
        /* Your program

        code goes here */
}
```



```
main()
{
        /* Your program

        code goes here */
}
```

All instructions and functions belonging to main are enclosed in curly brackets {…}. This is called 'block building'. In the above example, void means 'empty' and indicates that the main function does not require any input parameters and does not return any result after the instructions are executed. The two instances of the void keyword can also be omitted if desired.

Comments in C

All text strings and phrases that don't form part of the actual program are called 'comments'. Comments are ignored by the compiler, which means they do not occupy any space in memory. However, they are quite valuable for explaining the program (or important parts of the program) to other people. And of course, they're very useful for the author of the program as well. In many cases, you may not remember why you wrote the code in a particular manner when you look at your program several days later. And no matter how appropriate the saying 'lean is keen' may be in other contexts, it certainly doesn't apply to computer programs.

```
/*
Comments are
enclosed between
diagonal slashes
and asterisks. */
```

```
// This is a single-line comment.
```

Single-line comments begin with two diagonal slashes and end automatically at the end of the line.

A semicolon (;) is usually used to designate a comment in assembly-language programming, but in the C language it marks the end of an instruction.

#include

There are many declarations and functions that are not included in the ANSI standard for the C language, even though they may be necessary or very useful. They are commonly 'hidden' in libraries. You have to tell the compiler to include these library files, which are called 'header files', so it can use these declarations and functions when it compiles your C source code. You can recognise header files by the .h file extension.

Examples:

```
#include "stdio.h"
```

Used in C programs intended to be run on a PC.
[Standard input/output; includes the print function printf() .]

```
#include "sfr_r813.h"
```

The Renesas library. The names and bits of the registers of the R8C microcontroller, such as p1, pd1, p1_7 etc., are defined here.

Keywords in C

A total of 32 terms known as 'keywords' are defined in the ANSI standard for the C language. These keywords are reserved for the compiler. All keywords must be written in lower-case characters, and they are not allowed to be used for other purposes (such as naming variables).

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Many C compilers add supplementary keywords to those defined in the ANSI specification in order to make best use of the features of the compiler or microcontroller. The terms listed below are also designated as keywords for the R8C family of microcontrollers.

_asm	asm	near
_far	_Bool	restrict
_near	far	inline

Constants and Variables

Number systems

The C language can work with several different number systems (number bases): decimal, octal, and hexadecimal.
Numbers stated without any special identification (notation) are interpreted as decimal numbers by default. Numbers in all other number systems must be specially identified. Octal numbers begin with 0, hexadecimal numbers with 0x, and binary numbers with 0b.

Base	Notation	Available characters	Example
Decimal(10)	–	0123456789	5
Octal(8)	0…	01234567	05
Hexadecimal(16)	0x…	0123456789ABCDEF	0x5
Binary (2)	0b	0 1	0b11110000

The English (US) convention is always used for numerical notation. That means a full stop ('period') is used as the decimal marker for floating-point numbers. In C, the comma is used in as a separator in lists of numbers or variables. A colon marks a range of numbers.

Examples: **USA**

3.14159	
3,4	3 and 4
0:3	0->3, thus 0, 1, 2, and 3

Data types

In C, the type of a variable must be declared before it can be used. Otherwise the compiler will not know how much memory to allocate for the variable. Basically, you should always select a type that is adequate for the intended purpose and requires the least amount of memory space. The most important data types are listed below.

Type	**Memory space**	**Value range**
_Bool	8	0, 1
char	8	0 -> +255
signed char	8	-128 -> +127
int, short	16	-32768 -> +32767
unsigned int	16	0 -> +65535
long	32	-2147483648 -> +2147483647
float	32	-1.17..e-38F -> +3.4..e-38F

Examples:

```
_Bool   stop_button  // button has only two states: on & off
unsigned int       _year // 0 -> 65535 sufficient for year numbers
float    _volume      // floating-point number for calculations
```

Constants

Constants are numbers that cannot be changed in the program. That also includes all ‘normal’ numbers. Whole numbers (integers, or `int`) are written without a decimal marker (decimal point). Floating-point numbers (`float`) have a decimal point followed by additional digits. Characters (`char`) are enclosed between single quotation marks (`'`). Constants are declared using the `#define` keyword.

```
#define    <label>           value
```

No ; **because this is only relevant for the compiler**

Examples:

```
#define true 1                    // 1 = true
#define false 0                   // 0 = false
#define pi 3.14159                // the factor
#define letter_1 'A'             // 'A' key on the keyboard
```

> The names of constants, variables and functions can be freely selected, but they are not allowed to contain any keywords or operator symbols. Basically, only the letters of the English alphabet, numerals and the underscore (_) are used for names. You should chose names that give a good indication of the practical meaning of the constant. For instance, `alarm_btn` is much more meaningful than `t1` .

Variables

A variable is an item stored in memory that can be changed in the program. Variables can be numbers, letters, or text strings. In C, all variables must be declared before they can be used. Variables are considered to be ‘statements’, which means variable declarations must be terminated with a semicolon (**;**). Variables are defined as follows:

```
type        <label>           ;
```

; **because this is a processor instruction**

Examples:

```
_Bool        keypress ;
long         counter ;
float        radius ;
```

Variables are assigned values as follows:

```
<label>   =     value      ;
```

;

Examples:

```
keypress = 1 ;           min_val = counter - 50 ;
keypress = false ;       max_val = counter * counter ;
counter = 100 ;          _circum = radius * 2 * pi ;
```

!

Commands and instructions for the processor, which are called 'statements', are terminated with a semicolon (;).

Operators in C

Arithmetic operators

The symbols for arithmetic operators in C correspond to the familiar symbols used on pocket calculators:

```
+     addition              // examples:   y = x + 3 ;
-     subtraction           //             y = x – b ;
*     multiplication        //             y = a * b ;
/     division              //             Y = a / b ;
```

The equal sign has a different meaning in C than in ordinary mathematics. In C, it is called the ‘assignment operator’. That means the expression to the right of the equal sign is computed and the result is assigned to the variable to the left of the equal sign. The following expressions are thus allowed in C, but not in normal mathematics:

```
x = x+y ;        // compute x + y and store the result in x
x = -x ;         // change the sign of the variable x
```

Relational operators

Relational operators are used to compare variables. They return the result true or false , depending on the event.

```
>     greater than                ==    equal
>=    greater tnan or equal       !=    not equal
<     less than
<=    less than or equal
```

Logical operators

The logical operators AND, OR and NOT can be used to execute the familiar operations of digital logic.

		AND	OR	NOT
b	a	a && b	a || b	!a
0	0	0	0	1
0	1	0	1	0
1	0	0	1	1
1	1	1	1	0

Example:

```
if(_price <= max_price && _account > 1000)
        _buy();

/* The function _buy() will only be called if _price is less
than or equal to max_price and _account contains more than
1000 pounds */
```

The Renesas C compiler for the R8C has several additional logical operators that can be used for bitwise operations on variables:

&	for bitwise AND operations
|	for bitwise OR operations
^	for bitwise XOR operations

b	a	a^b
0	0	0
0	1	0
1	0	0
1	1	1

Examples:

```
a   = 10011010        a   = 10011010        a   = 10011010
b   = 11000011        b   = 11000011        b   = 11000011
a&b = 10000010        a|b = 11011011        a^b = 01011001
```

Shortcuts

Americans are masters at inventing shortcuts. That's especially true for Dennis Ritchie and Brian Kernigham, the inventors of the C language. Many programmers type in their programs using the 'hunt-and-peck' system, so they try to avoid any unnecessary typing effort.

Shortcut	Normal	Shortcut	Normal
a*=b	a = a*b	a<<=b	a = a<<b
a/=b	a = a/b	a>>=b	a = a>>b
a+=b	a = a+b	a&=b	a = a&b
a-=b	a = a-b	a|=b	a = a|b
a%=b	a = a%b	a^=b	a = a^b
a++	a = a+1 (increment)		
a--	a = a-1 (decrement)		

Example:

```
for(t=0, t<100000, t++);        /* timer loop */
```

Functions in C

The function concept

Functions are the essence of the C programming language. A function can be called from the main routine or from any other function. Every C program must include at least one function, which is called main() . It is automatically called when the program is started.
Functions are individual program segments (blocks) that perform specific activities (operations), such as the familiar operations performed by pocket calculators:

CE clears the input memory of a pocket calculator. In C, it would be described as a function that does not require any numerical input (parameters) and does not return any sort of number.

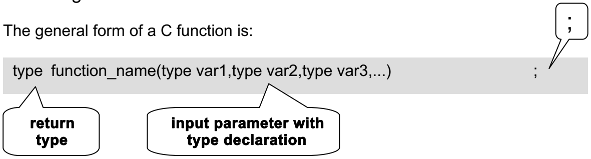
1/x expects to receive a number as input in order to calculate its inverse value. In C, this is described as a function with an input parameter.

+ - * / , by contrast, requires two parameters as input, and O , the summation function, requires several parameters.

sin and log , on the other hand, are functions that require one input parameter.

Declaring a function

The general form of a C function is:



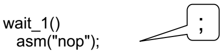
```
type  function_name(type var1,type var2,type var3,...)                    ;
```

Examples:

A function with no input or output parameters:

```
void wait_1(void)
    asm("nop");             //  Call    no  operation in assembly language
```

The word void tells the compiler that the function wait_1 does not require any input parameters and does not return any result. The void keywords can also be omitted:



```
wait_1()
  asm("nop");
```

If a function contains more than one instruction, the instructions must grouped into a block by enclosing them in curly brackets ({ }). In that case the semicolon at the end must be omitted.

no ;

```
wait_2()
{
        asm("nop")        // Wait three times
       asm("nop")
        asm("nop")
}
```

no ;

A function with an input parameter but no return parameters:

```
int t;

wait_3(int several_times)
  for(t=0;,t<= several_times;t++);
```

/* The for loop increases the value of t stepwise
(increments t) starting from 0 until it reaches the
value of the input several_times (time delay) */

A function with input parameters and a return parameter:

```
float _volume(float length, float width, float height)
  return length*width*height;
```

/* This function expects three inputs, which are stored
in the variables length, width and height. The product
of these three variables is then calculated and returned
to the caller as a floating-point number */.

Calling a function

Functions are called by simply stating their names. This can be done at any location in the program. After the function has been processed, which can be recognised by the ; in case of a function containing only one instruction or the curly bracket } in case of a function containing several instructions, a return to the calling location occurs automatically. The keyword return has a different meaning in C than in assembly language. In C, it designates the return value instead of the end of a subroutine or function.

Multiple-level function nesting is allowed. 'Function nesting' means that one function calls a second function, which in turn calls a third function, and so on.

Examples:

Calling a function from the main routine without any input or return parameters:

```
void  main(void)
{
        wait_1();
}
```

Calling a function from the main routine with an input parameter but no return parameters:

```
void main(void);
{
        wait_3(100);
}
```

// The constant 100 is passed to function wait_3   .

Calling a function from the main routine with input and return parameters:

```
void main(void);
{
        no_of_litres = _volume(a,b,c);
}
```

/* The values of the variables a, b, and c are passed to the
function _volume. The function then calculates the volume of
a body and returns the result to the variable no_of_litres.

Program Control

if

It frequently happens that an instruction or block of instructions should only be executed if a certain condition is satisfied. A condition is satisfied if a test of the condition returns the value true . Every number except zero is regarded as true ; zero is regarded as false .

The general form is:

;

```
if    (condition)      statement  ;
```

If several instructions are to be executed when the condition is satisfied, they must be grouped into a block.

no ;

```
if (condition)
{
        statement_1
        statement_2
        statement_3
        // . . .
}
```

no ;

Examples:

```
if(button == 3)
        red_led = _on;
```

```
if(button == 3)
{
        grn_led = _off;
        red_led = _on;
}
```

if...else

If one instruction or block of instructions is to be executed when a condition is satisfied, while an another instruction of block of instructions is to be executed when the condition is not satisfied, an if...else conditional statement is used.
The general form is:

; ;

```
if    (condition)      statement_1 ;    else     statement_2 ;
```

If several instructions are to be executed, they must be grouped into a block.

Examples:

```
if(button == 3)
       red_led = _on;
else
       grn_led = _on;
```

;

;

```
if(button == 3)
{
       grn_led = _off;
       red_led = _on;
}
else
       grn_led = _on;
```

==

block

markers

```
if(button == 3)
{
       grn_led = _off;
       red_led = _on;
}
else
{
       grn_led = _on;
       red_led = _off;
}
```

switch

If a conditional statement has more than one or two possible outcomes, it is very tedious to implement it using the if...else structure. In that case it's better to use the switch /case structure with multiple alternatives. This type of program control can be compared to a rotary selector switch with multiple positions (cases).

The general form is:

```
switch     (variable)
{
       case        constant_1 ;
                   instruction_1;
                   break;

       case        constant_2;
                   instruction_2
                   break;

       case        constant_3;
                   instruction_3

       case        // . . .
                   break;

       default     instruction_x;
}
```

no ;

The switch function compares the content of variable to the value of a constant (constant_x) for each of the defined cases (case). If the result of the comparison is positive, the corresponding instruction (instruction_x) or block of instructions is executed. Program control returns to the caller of the case statement when the break keyword is reached. If none of the cases listed in the case statement is found, the instruction following default is executed. The default portion can be omitted if it is not necessary.

Example:

```
            switch      (_button)
            {
                  case        1:
                              red_led = _on;
                              break;

                  case        2:
                              yel_led = _on;
                              break;

                  case        3:
                              grn_led = _on;


                  default     blu_led = _on;
            }
```

block

markers

may be omitted

/* Enable the red LED if the value of _button = 1, the yellow
LED if it is 2, and the green LED if it is 3. If the variable
_button does not contain 1, 2, or 3 (e.g. 4), enable the blue
LED */

for

A for loop is used if part of a program must be executed multiple times.

The general form is:

```
for(start_value; end_condition; step_size)
       instruction_1;
```

;

When the for loop is called, start_value is assigned to a previously defined count variable. The count variable is then incremented or decremented by the value of step_size each time the loop is executed, until the test of end_condition yields the logical value true .
If several instructions are to be executed, curly brackets ({ }) must be used to group them into a function block.

Examples:

```
int   t;

for(t=0, t < 10, t++)
       blink_led();
```

/* Integer variable t is assigned the value 0 when the for loop is entered. Next, the function blink_led() is called. After the first pass through the loop, the variable t is incremented (t++) to the value 1. As 1 is less than 10, the process continues until t = 9. The loop is thus executed ten times. */

```
a = 2;            variables
b = 10;
c = 4;
                  count variable
int i;

for(i = a; i < b; i+ = c)
{
       red_led = _on;
       red_led = _off;
}
```

/* This loop is executed only two times. */

while

A while loop is used when execution is tied to a condition.

The general form is:

```
                          no ;
while(_condition)
{
       instruction_1;
       instruction_2;
       // . . .
}
       no ;
```

When the while loop is called, the value of the condition (_condition) is first tested. If the result is positive (true), the instructions (instruction_x) are executed repeatedly until the result of the test is false.

Example:

```
#define true 1
```

==

```
while(i == true)
{
        i = button_pressed();
        blink_led();
}
```

/* The function blink_led is executed until the function button_pressed no longer returns the value 1 to variable i.

do...while

A while loop will not be executed if the condition is not satisfied at the beginning of the loop. If it is necessary for instructions to be executed at least once, the condition must be tested at the end instead. A do...while loop is used in such cases.

The general form is:

no ;

```
do
{
        instruction_1;
        instruction_2;
        instruction_3;
        // . . .
}
while(_condition);
```

no ;

;

Example:

```
#define true 1

do
{
        i = button_pressed();
        blink_led();
}
while(i == true);
```

==

/* The function blink_led is executed at least once */

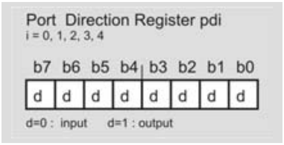
Header file sfr_r813.h

The header file sfr_r813.h provides access to the special function registers (SFRs) of the R8C microcontroller. These registers contain the basic settings for the microcontroller, such as the port directions (in/out), timer settings, A/D converter settings, UART settings, and so on.
The R8C microcontroller has more than 50 SFRs. That means we have to limit ourselves here to a selection of the most important SFRs. Refer to the **R8C/13 Group Hardware Manual** for detailed information on all of the SFRs.

Port registers (P0, P1, P2, P3 and P4)

A port is a memory location that is connected to the pins of the microcontroller and is thus externally accessible. Ports are used for inputting and outputting data. A port can be either an input or an output. When the microcontroller is started up, all ports are configured as inputs by default. The direction (input or output) can be changed using the Port Direction (PD) registers.

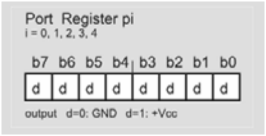
The following example shows how to change the port direction:



Examples:

```
pd1 = 0x0F;
/* port1, bits 0:3 = output
   bits 4:7 = input */

pd2_3 = 1;
/* port2,bit3 = output */
```

… and the following example shows how to output data via a port:



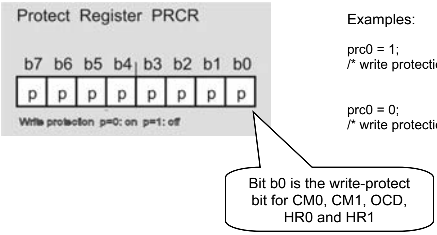
Examples:

```
p1 = 0x0F;
/* port1, bits 0:3 = 1
   bits 4:7 = 0 */

pd2_3 = 0;
/* port2,bit3 = 0 */
```

/* If a 1 is written to a port register, the supply voltage (e.g.,
+5 V) will be present on the corresponding pin. A 0 causes the pin
to be at ground potential (0 V) */

Protection registers (PRCR)

The PRCR registers can be used to protect the contents of other important registers against overwriting (if the program goes out of control, for instance).

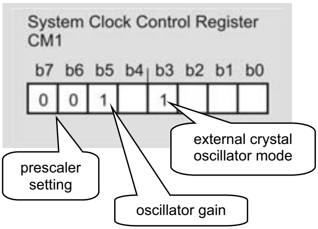

Protect Register PRCR
b7 b6 b5 b4 b3 b2 b1 b0
p p p p p p p p
Write protection p=0: on p=1: off
Bit b0 is the write-protect bit for CM0, CM1, OCD, HR0 and HR1

Examples:

```
prc0 = 1;
/* write protection disabled */

prc0 = 0;
/* write protection enabled */
```

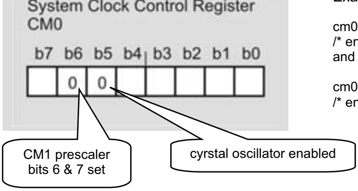
System Clock Control registers (CM1, CM2 and OSD)

The R8C microcontroller has two oscillators to provide the clock for the CPU. One oscillator is internal and is called ‘on-chip oscillator’, while the other is external and is called ‘main clock’. The ‘main clock’ oscillator uses a quartz crystal connected to the Xin and Xout pins.
The CM registers determine how the microcontroller clock signal is generated. In addition, a prescaler can be enabled and configured to reduce the frequency of the processor clock.


System Clock Control Register CM1
b7 b6 b5 b4 b3 b2 b1 b0
0 0 1 1
external crystal oscillator mode
prescaler setting
oscillator gain

Examples:

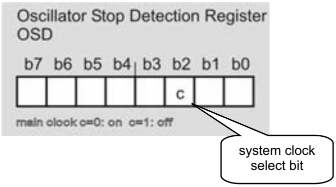
```
cm13 = 1;
/* Xin/Xout on ports p46 and p47
= ext. xtal */

cm15 = 1;
/* Xin/Xout driver high */

cm16 = 0;
/* prescaler for CPU clock */

cm17 = 0;
/* prescaler for CPU clock */
```


System Clock Control Register CM0
b7 b6 b5 b4 b3 b2 b1 b0
0 0
CM1 prescaler bits 6 & 7 set
cyrstal oscillator enabled

Examples:

```
cm05 = 0;
/* enable Xin/Xout on ports p4.6
and p4.7 */

cm06 = 0;
/* enable prescaler */
```

The Oscillator Stop Detection (OSD) register is used together with the other registers to select the clock source, and it is also responsible for monitoring the clock signal.


Oscillator Stop Detection Register
OSD
b7 b6 b5 b4 b3 b2 b1 b0
c
main clock c=0: on c=1: off
system clock select bit

Examples:

```
osd2 = 0;
/* enable external xtal osc. as
CPU clock source */

osd2 = 1;
/* disable external xtal osc. */
```

Header file math.h

The header file math.h contains the library of mathematical functions for the R8C. Here we describe some of the most important functions.

Functions of type double
f_name(double x);

sin();	sqrt();
cos();	
tan();	exp();
	log();
asin();	log10();
acos();	
atan();	mod();
sinh();	fabs();
cosh();	floor();
tanh();	ceil();

Functions of type float
f_name(float x);

sinf();	sqrtf();
cosf();	powf();
tanf();	
	expf();
asinf();	logf();
acosf();	log10f();
atanf();	
sinhf();	fabsf();
coshf();	floorf();
tanhf();	ceilf();

Functions of type double
f_name(double x, double y);

pow();
fmod();
atan2();
fmod();

Functions of type float
f_name(float x, float y);

powf();
atan2f();
fmodf();