The Complete Source for Electronics: PROJECTS, TECHNOLOGY and EMBEDDED

DSP Course continued!

June 2011

+ USB workings demystified

elektor

www.elektor.com

DIY Radiation Detector

Measure Gamma Rays with a Photodiode

Developing Apps for Android

Get out your PC, BeagleBoard & Smartphone or Tablet

+ VGA Add-on for Microcontrollers

+ Geolocation with the ATM18

US $ 7.95 - Canada $ 7.95

HUGE discounts

on select RIGOL equipment

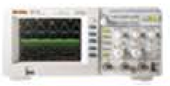
- for a limited time only!

DG1022 - 20MHz Function/Arbitrary Waveform Generator

Was $695 Now only $499

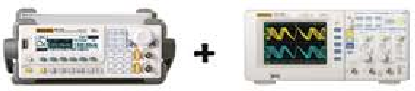
DS1052D - 50MHz Mixed-signal Scope 2ch + 16 Digital Channels

Was $1,095 Now only $899



DS1102D - 100MHz Mixed-signal Scope - 2ch +16 Digital Channels

Was $1,495 Now only $1,099



DG1022 + DS1052E
Basic Signal Analysis Package

20MHz Function/AWG +
50 MHz 2-Ch Digital Oscilloscope

Was $1,094 Now only $799

DS1102CA + DG2041A
Standalone Signal Analysis Package

40MHz Function/AWG +
100 MHz 2-Channel Digital Oscilloscope

Was $2,290 Now only $1,799

Hurry! This sale is only for a limited time!
Call us at 888-772-3544
or go to www.saelig.com today!

FREE Multi-Function Tool with your order of $50 or more!

Type "Multi-Function Tool" in the search box on our website

Saelig.com unique electronics Follow us:

do more wire less

Wixel

USB.
Wireless.
Programmable.
All for only $19.95.

Introducing the Wixel, a programmable microcontroller module with integrated USB and a 2.4 GHz radio. Write your own program or load pre-compiled, open-source apps to give your next project a wireless serial link, create a remote sensor network, and so much more.

www.pololu.com/wixel

actual size

Pololu
Engage Your Brain

If you need more power, just switch over from PIC or dsPIC to PIC32 easily. We have been carefully planing backward compatibility for over 500 of our library functions, so you will be able to literally copy-paste your existing codes and build them with just a few adjustments.

MikroElektronika
DEVELOPMENT TOOLS | COMPILERS | BOOKS

www.mikroe.com

Poor man's radiation meter

After the terrible accident at the Fukushima nuclear power plant the demand for Geiger counters surged. No wonder as many people wanted to establish fallout levels in their own area. A company like Conrad Electronics with several types of Geiger counter in their catalog saw their stock vanish within a few days. The manufacturer of the devices having problems coping with the demand is evidenced by Conrad Electronics on their website advising that supplies are likely to slip into February 2012!
For Elektor the sudden interest in Geiger counters was reason enough to start thinking about a simple, inexpensive and homebrew radiation detector suitable for all measurements in this field. After all, radiation is permanent and everywhere, and possible sources should be identified carefully, as well as the levels of radiation they put out.
It's a long known fact that a simple photodiode can be used as a radiation detector if properly set up. When combined with a low-noise amplifier, you have a basic instrument for making the presence of X-rays and gamma rays visible and audible. Although it does not allow you to measure absolute values, you will get a reasonable idea of the radiation fields surrounding you. So be sure to read and digest the article "Measure Gamma Rays with a Photodiode" on page 22.
Another 'hot' topic, but in a slightly different meaning, is the Android operating system running in numerous smartphones and other portable equipment. In this edition we show you how to develop Android apps for your own applications in a relatively simple manner. A good deal of attention is given to the relevant hardware and software. Also highly recommended!
Have fun with these and many other articles in the June 2011 edition. I'm already busy editing and compiling the articles to go into the upcoming July & August 'Project Generator' edition!

Jan Buiting, Editor

6 **Colophon**
Who's who at Elektor magazine.

8 **News & New Products**
A monthly roundup of all the latest in electronics land.

14 **Elektor OSVP¹**
Introducing a state of the art electrical self-balancing vehicle that's open-source to a high degree.

16 **Developing Apps for Android**
Here we explore how to build an App for Android-based smartphones with no more than a TI Beagle board and a PC.

22 **Measure Gamma Rays with a Photodiode**
A highly topical how-to on the use of a humble photodiode type BPW34 to detect X-rays and gamma radiation.

28 **Picoscope 3000 on Test**
We put a Picoscope 3206B through its paces and examined its hardware and software complement.

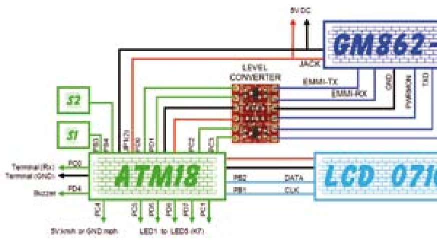
32 **Geolocation with the ATM18**
There's no end to what you can do with an Elektor ATM18 module. And now you can find its whereabouts anywhere on the globe.

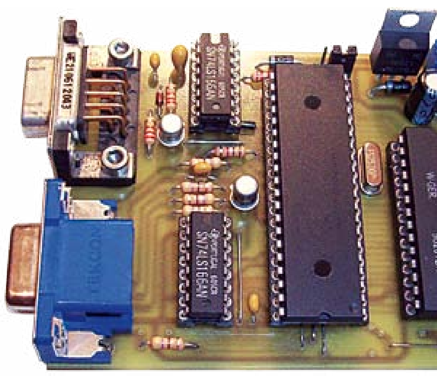
38 **Audio DSP Course (2)**
In this the second part of the course we tackle the process of DSP programming.

43 **E-Labs Inside: PCB design: beware the tiny details!**
The lab baffled by 15 thousands of a millimeter, really!

44 **E-Labs Inside: Hurricane at SMD scale**
Taming a hot air reflow solder station to make it suitable for those tiny parts.

45 **E-Labs Inside: Down to Earth**
Ruminations on the use of a device to tackle Phase and Earth issues in AC wall outlets.





CONTENTS

Volume 3
June 2011
no. 30

16 Developing Apps for Android

Current estimates indicate that 350,000 new Android phones go on the air each year. What is the reason for this amazing success? Is it Google? Or the fact that it's open source? Or because it works well? In any case, the specific reason isn't that important; what matters is that you can also join the trend and develop your own Android apps. Here's how!

22 Measure Gamma Rays with a Photodiode

Geiger-Müller counter tubes are getting hard to find and expensive, and even if you do manage to get hold of one, you will still need to find a way to generate its operating voltage of several hundred volts. It is less well known that even a humble photodiode such as the BPW34 can be used to detect X-rays and gamma radiation.

32 Geolocation with the ATM18

A GSM modem module with a built-in GPS receiver allows its location to be determined very precisely. If you combine it with an Elektor ATM18 module and install the lot in your car, you have a tracking device that can send you e-mail or text messages to let you know exactly where your prize vehicle is.

62 VGA Add-on for Microcontrollers

Many projects require a large amount of information to be displayed, but the size of the display itself is often a problem. One solution is to use an old 14" or 15" computer monitor that's been scrapped but is still working. The VGA board described in this article lets you do just this, and is compatible with any microcontroller that has a serial port.

46 **E-Labs Inside: Driver plate modification for ElektorWheelie**
Strengthening the wheel bolts is necessary in some cases. Here's how.

48 **Here comes the Bus! (6)**
The design has ripened to the extent that a real PCB can be presented to start experimenting for real.

54 **Inside USB**
This is for all of you happy with all the USB pluggin' going on but really wanting to know what's going on under the bonnet.

58 **E-blocks: Flowcode RC5**
E-blocks and Flowcode make a great team when you want to add infrared remote control to your projects!

62 **VGA Add-on for Microcontrollers**
Got an old computer monitor? Put your microcontroller on the screen with this clever board operating on serial commands.

68 **Design Tips: AVR and 8051 drive 2-color LEDs**
Combine software and hardware to control 2-color LEDs in ingenious ways.

71 **Stellaris ARM Controller goes Biological**
Design considerations for an egg incubator made from a rice cooker. Microcontroller-driven, of course.

74 **Gerard's Columns**
Those Lying Instruments

75 **Hexadoku**
Elektor's monthly puzzle with an electronics touch.

76 **Retronics: Solid Light: the Remarkable Prehistory of the LED**
Regular feature on electronics 'odd & ancient'. Series Editor: Jan Buiting

84 **Coming Attractions**
Next month in Elektor magazine.

elektor | international media bv

Elektor International Media provides a multimedia and interactive platform for everyone interested in electronics. From professionals passionate about their work to enthusiasts with professional ambitions. From beginner to diehard, from student to lecturer. Information, education, inspiration and entertainment. Analog and digital; practical and theoretical; software and hardware.

The Complete Source for Electronics: PROJECTS, TECHNOLOGY and EMBEDDED
DSP Course continued!
+ USB workings demystified
elektor
DIY Radiation Detector
Measure Gamma Rays with a Photodiode
Developing Apps for Android
Get out your PC, BeagleBoard & Smartphone or Tablet
+ VGA Add-on for Microcontrollers
+ Geolocation with the ATM18
ANALOG • DIGITAL
MICROCONTROLLERS & EMBEDDED
AUDIO • TEST & MEASUREMENT

No. 30, JUNE 2011 ISSN 1947-3753

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

Elektor (ISSN 1947-3753) is published monthly (except for one issue in July/August) at $39.95 per year, Canada add $11.00 per year;
by Elektor International Media LLC,
4 Park Street, Vernon, CT 06066, USA.
Phone: 860-875-2199, Fax: 860-871-0411.
www.elektor.com

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

International Editor:
Wisse Hettinga (w.hettinga@elektor.com)

Editor: Jan Buiting (editor@elektor.com)

International editorial staff: Harry Baggen, Thijs Beckers, Eduardo Corral, Ernst Krempelsauer, Jens Nickel, Clemens Valens

Design staff: Christian Vossen (Head), Thijs Beckers, Ton Giesberts, Luc Lemmens, Jan Visser

Graphic design / DTP: Giel Dols, Mart Schroijen

Publisher: Hugo Van haecke (h.vanhaecke@elektor.com)

Marketing: Carlo van Nistelrooy

Customer Services: sales@elektor.com

Subscriptions:
Elektor US, 4 Park Street, Vernon, CT 06066, USA.
Phone: 860-875-2199, Fax: 860-871-0411
Internet: www.elektor.com
E-mail: subscriptionsus@elektor.com

Elektor Proton Robot

New

A versatile platform for learning and experimenting

The Proton Robot from Elektor is a versatile platform that's suitable for students, enthusiasts and professionals alike. The robot can operate with a variety of microcontroller families and it supports a broad spectrum of sensors and actuators. This is an ideal opportunity to invest in a unique combination of knowledge and fun!

Characteristics

- Ultrasonic distance sensor
- 8 LEDs in the mouth
- 8 LEDs in the body
- Piezoelectric speaker
- 3 infrared distance sensors
- Motor drive module
- 3 line detectors
- LED eyes
- 2 phototransistors
- 2 servomotors
- LCD
- Red and black buttons
- Audio module
- Gripper

Ordering

You can order the robot ready assembled and tested but also as a complete kit for DIY assembly.

Complete kit :

Body, head, audio, gripper and PIC or AVR control board to choose

$1745.00

Ready assembled and tested robot:

Body, head, audio, gripper and PIC or AVR control board to choose

$2375.00

Further information and ordering at

www.elektor.com/proton

Head Office:
Elektor International Media b.v.
PO Box 11 NL-6114-ZG Susteren The Netherlands
Telephone: (+31) 46 4389444, Fax: (+31) 46 4370161

US Advertising:
Strategic Media Marketing, Peter Wostrel,
1187 Washington St., Gloucester MA 01930 USA.
Phone: 978-281-7708, Fax: 978-281-7706
E-mail: peter@smmarketing.us
Advertising rates and terms available on request.

FreeRTOS Seven released

FreeRTOS™, an open source embedded RTOS with implementations on more than 27 embedded architectures, was first released over eight years ago. Over the years it has become very popular and now racks up more than 7000 downloads per month, making it one of the most widely used embedded operating systems in the world. The license allows users to deploy applications without publishing their proprietary application code.

The latest release of FreeRTOS, Version 7.0, features a new software timer implementation allows timer callbacks to be executed in a task context instead of an interrupt context, which does not consume any processing time or generate any run time overhead unless a timer has actually expired. FreeRTOS Seven also benefits from improved performance and two new demonstration projects. The kernel is fully backward compatible with the previous generation, and users who upgrade will not need to change any of their own code. Upgrading from version 6.x is as simple as dropping the new kernel source files into your project.

A Texas Instruments MSP430X MCU Code Composer Studio port has been added, bringing the list of official ports to 52, distributed over 27 MCU architectures. Each official port includes a documented pre-configured example application that demonstrates the kernel features to accelerate the learning curve and learning and facilitate out-of-the-box product development.

FreeRTOS is a portable, open source, royalty free, mini real time kernel, which is free to both download and deploy. It can be used in commercial applications and does not require proprietary source code to be exposed. A commercial license and support upgrade path are also available from High Integrity Systems, the global engineering partner company behind licensing and support for OpenRTOS and the safety critical MCU operating system SafeRTOS.

Downloaded more than 150,000 times during 2009–2010, FreeRTOS is among the most widely downloaded and used real time operating systems. In March 2010 it topped a list of the industry's favorite RTOS kernels (pbli.tk/RTOSsurvey), based on an independent survey of embedded system engineers asked to select the operating systems being considered for use.

http://www.FreeRTOS.org (110422-I)

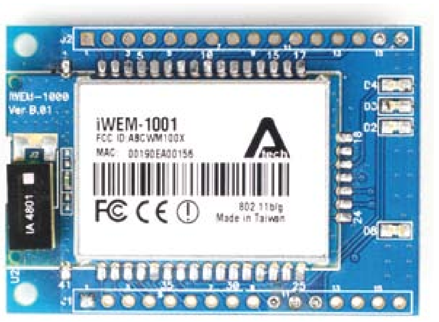
iWEM-1001 Wi-Fi modules target M2M links

The iWEM-1001 series of Wi-Fi modules from Lemos International are designed for industrial applications and feature ultra low power with fast boot-up and connection capability. They are key products for completing the last mile of M2M wireless connectivity.

iWEM-1001 Wi-Fi modules are fully integrated Wi-Fi solutions in a compact IC-style SMT package. They enable engineers to give any desired product Wi-Fi connectivity without requiring RF design experience or months of hardware and software development effort. The only hardware requirement for the host system to enable the addition of Wi-Fi connectivity is a simple three-wire UART interface, which is a standard feature of most microcontrollers.

The iWEM-1001 power saving architecture operates in two modes: active mode and sleep mode. Current consumption in the active mode is 30 to 200 mA, depending on TX data density. It drops to less than 0.5 mA in sleep mode, so users can maximize battery life by keeping the iWEM-1000 in sleep mode most of time and waking it up only when it is needed.

Module features include Wi-Fi 80211b/g PHY/MAC up to 54 Mbps; easy 3-wire interfacing to any desired MCU; choice of embedded chip, UFL or SMA antenna; universal sensor interface for monitoring pushbuttons, temperature, motion and acceleration; 11 GPIOs; pre-installed Atech Wi-Fi Thin Client application firmware for full networking functionality; support for open, WEP-40, WEP-128, WPA-PSK and WPA2-PSK security, TCP/IP, DHCP, UDP, DNS, ARP and ICMP; module configuration using a simple ASCII command set, and over the air TFTP firmware upgrade capability.



Potential applications include industrial and home automation, wireless sensing and control, wireless smart metering, cable replacement, asset tracking, health and fitness monitoring, and MCU interface connectivity.

www.lemosint.com (110422-II)

You've got a great idea – but is it patentable?

If you have an idea for an invention but you aren't sure whether it is truly unique and possibly eligible for patent protection, what can you do? The conventional approach is to enlist the services of a patent specialist, who looks for related existing patents and tells you whether your idea has already been patented. This may involve a substantial investment, depending on the complexity of your idea, and it requires that you disclose your idea to someone else. Although it's unlikely that the person performing the patent search for you will try to steal your idea, it's understandable that you want to reveal your idea to as few people as possible.

MIDWEST PATENT SERVICES, LLC
SEARCH
Complete A to Z Guide to Performing A Patent Search
Entrepreneurship

Another drawback of this approach is that the search is performed only once. If you aren't ready to file your application right away, other relevant applications may be filed in the meantime. Also, many inventors like to search existing patents before deciding to apply for a patent, since this gives them a feeling for whether their invention is sufficiently novel.

Now Midwest Patent Services offers you an alternative, in the form of a DVD that walks you through the patent search process step by step and allows you to follow along on your computer. Three databases – the United States Patent Office website, Google Patents and FreePatentsOnline – are covered in detail to enable you to perform a thorough patent search. The DVD allows you to conduct patent searches without revealing your idea to anybody, and it comes with a quality guarantee.

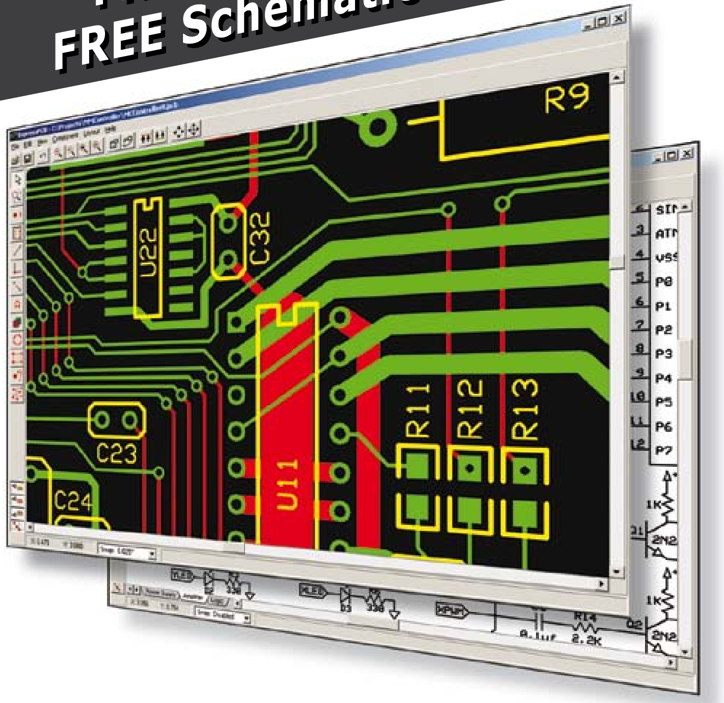
Visit the Midwest Patent Services website to learn more about the DVD and place an order.

www.midwestpatentservices.com (110422-III)

Advertisement

$51 For 3 PCBs

FREE Layout Software!
FREE Schematic Software!



01 DOWNLOAD our free CAD software
02 DESIGN your two or four layer PC board
03 SEND us your design with just a click
04 RECEIVE top quality boards in just days

expresspcb.com

PAR30 12 W LED spotlights feature integrated dimming

FZLED, a manufacturer of high-performance LED lighting products, has released a new line of PAR30 12 W LED spotlights with a lifetime of more than 35,000 hours. Users can choose from warm white (3000K CCT) or cool white (6000K CCT) versions, which provide luminous fluxes of 550 or 700 lumen, respectively. Featuring an excellent cost/performance ratio, these environmentally friendly, high-performance PAR30 LED spotlights are made to FZLED's exacting quality standards and offer consumers exceptional value.

FZLED 12-W PAR30 energy efficient LED spotlights have an AC input voltage range of 90–264 V and can be fitted directly in standard E26/E27 sockets, and they offer a choice of 55° and 120° beam angles. The built-in dimming function allows the spotlights to operate at 12%, 25%, 50% or 100% of full intensity. The desired brightness level is easily selected by operating the power switch several time in succession.

Applications include indoor lighting, architectural lighting, floodlighting, mood lighting, and biological lighting. The lamps are available in dimmable and non-dimmable versions and feature low energy consumption, zero emission of heat, UV and IR, long lifetime, high driver efficiency (better than 80%), a power factor greater than 0.8, and compatibility with E26/E27 sockets. The devices are CE, FCC and ETL approved.

FZLED 12-W PAR30 LED spotlights are currently available in Taiwan and Singapore, and FZLED is keen to establish relationships with other distribution partners in order to provide consumers around the world with their innovative, energy saving and high-performance lighting solutions.

www.fzled.com.tw (110422-IV)

Wireless audio distribution leverages DECT technology

Cambridge Consultants have launched a new wireless audio distribution system for hearing assistance or simultaneous translation in auditoria and conference centers. Dubbed Salix, the system uses a Digital Enhanced Cordless Telecommunication (DECT) platform to eliminate the high installation and configuration costs of current infrared systems, delivering high quality stereo audio at a fraction of the installed cost of an infrared system.

Most audio distribution systems in use

today are based on infrared technology, which has limited range and requires a clear line of sight between the transmitter and receiver, which requires the installation of several transmitters in a venue, generally on the ceiling. This can be expensive, not only because of the need for multiple transmitters, but also because of the time-consuming and complicated installation. Salix overcomes these problems by utilizing DECT wireless technology, which has an effective operating range of at least 100 meters, does not require line of sight contact, and is self-configuring to ensure interference-free transmission. The result is high-quality, zero-dropout audio distribution from a single transmitter installed anywhere – significantly reducing hardware and installation costs. For simultaneous translation applications, several Salix systems can coexist in an auditorium, conference centre or school, thanks to a robust spectrum etiquette scheme shared by all users of the DECT bands.

The Salix system comprises a transmitter board that can be populated for one or two stereo channels and a receiver board with selector buttons for volume, power and audio program, an audio output socket and a built-in lithium polymer battery. A modern high quality, low latency music codec delivers stereo audio with a 15 kHz bandwidth.

The Salix reference design, which has been tested and proven, is available as a hardware documentation package including photoplot and assembly information, with executable software for the transmitter and receiver modules. Source code licensing is also available for custom design.

www.cambridgeconsultants.com (110422-VI)

Wireless IEEE 802.11.g sensor family operates for years on 1.5 V AA batteries

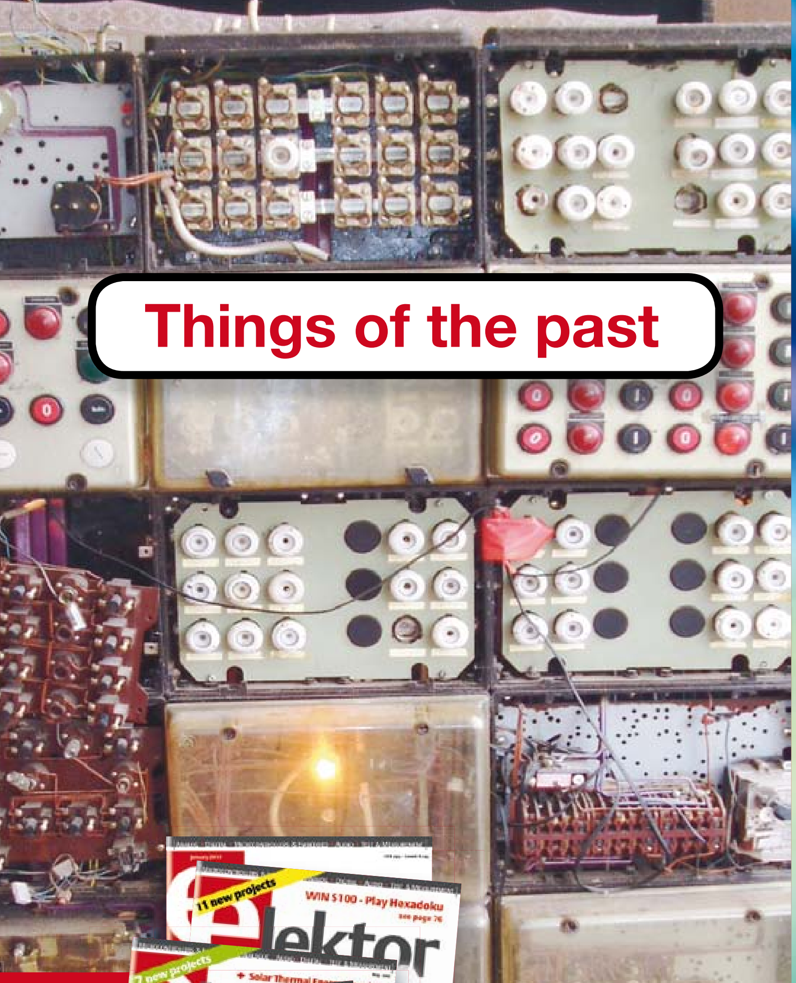
Point Six Wireless has launched its next generation of ultra low power 802.11 sensors. The product line consists of temperature, humidity, thermistor, RTD, pulse counter and analog input modules with integrated IEEE 802.11.g radios supporting WEP 128 and WPA2 security. All products include on-board data logging in nonvolatile memory to provide backup in case of network outage. Local audible and visual alarms alert personal to problems at the point of measurement, which are also reported over the WiFi network. The sensors are designed to work with industry standard 802.11 access points for easy installation. This capability eliminates the need for repeaters, controllers and coordinators typically required in most wireless system

deployments. The 802.11 integration strategy results in lower installation costs and on-going maintenance expenditures than the usual overlay strategy of installing another wireless network where one already exists. Designed to leverage the proliferation of 802.11 infrastructures in health care, education, food service and industrial facilities, these products enable IT managers to derive added value from their WiFi network structures.

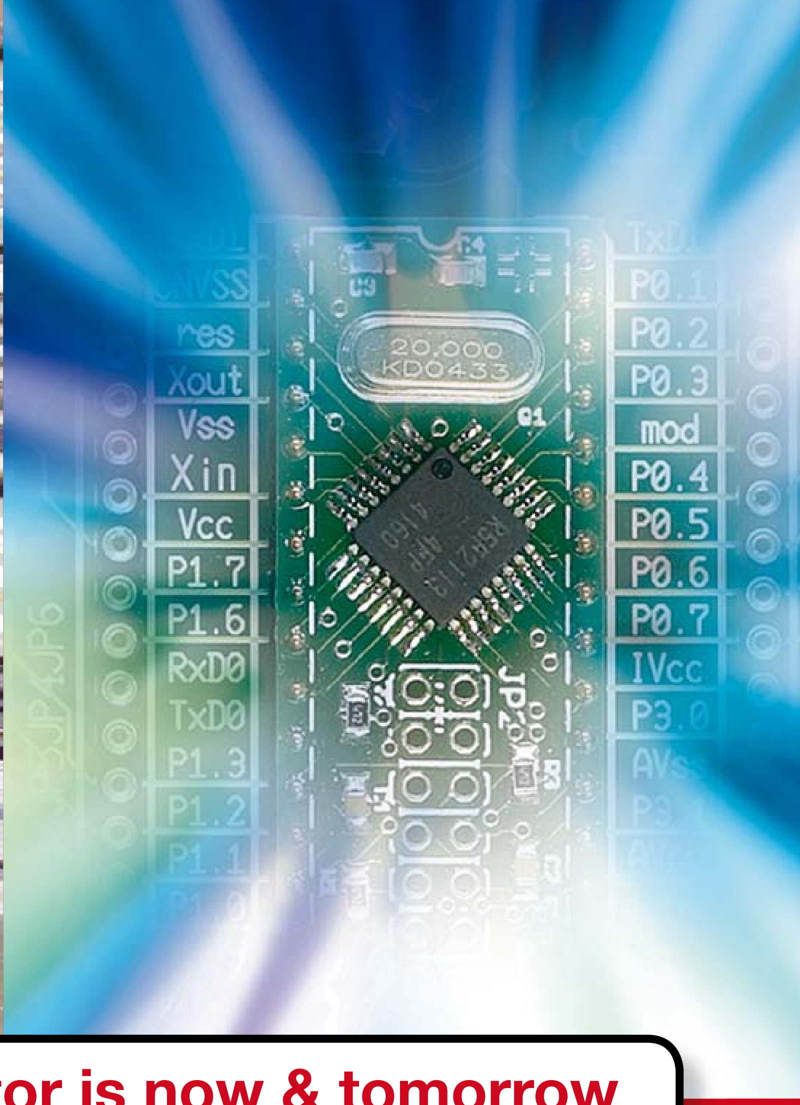
www.pointsix.com (110422-V)

LUXEON A Lumileds boast illumination-grade specs

The recently launched LUXEON A product line is the latest series of illumination grade LEDs from Philips Lumileds. They reduce the engineering effort required for developing new products and deliver freedom from binning. LUXEON A LEDs share the LUXEON Rebel ES platform and footprint and incorporate a 2-mm^2 thin film flip chip and Lumiramic phosphor technology to deliver superior quality illumination at 2700 K or 3000 K with very high efficiency and light output.



Things of the past



Elektor is now & tomorrow

Secure a head start in electronics with a subscription!

11 Issues including the summer double issue for just $39.95*

Advantages to subscribers

- Cheaper than 11 issues from the newsstand: **Save 57%** off the cover price of $92.45
- With every issue subscribers get up to **40% discount** on selected Elektor products
- No lines, travelling, parking fees or 'sold out'; Elektor is mailed to you every month
- Always up to date – read your copy before everyone else

* Offer available in US and Canada only. Canada add $11 per year.

VGL-USA-1

www.elektor.com/usa • Phone 860-875-2199

Or use the subscription order form near the end of the magazine.

elektor

All LUXEON A LEDs are hot-tested and specified at a junction temperature of 85 °C, which represents real-world operating conditions. Lumileds' unique chip and phosphor technology enables color targeting to ensure that all LUXEON A emitters fall within a single three-step MacAdam ellipse on the black body curve. The quality, uniformity, and consistency of the light from LED to LED eliminates the long-standing reservations many luminaire manufacturers and lighting designers have regarding the suitability of LED technology for illumination tasks.

Freedom from binning gives the industry high confidence in light quality, and Lumileds' hot testing pays dividends for engineers who design LED products. For a long time the LED industry tested and specified products at an LED chip junction temperature of 25 °C, even though it's common knowledge that LED junction temperatures close to 85 °C are more realistic in applications such as down lights and retrofit lamps. As a result, luminaire manufacturers had to perform many complex calculations to determine the actual light output, efficiency and color points of their products. Lumileds' hot testing simplifies this process by providing data for actual operating conditions, thereby eliminating ambiguity, exaggerated performance claims and unnecessary engineering effort.

www.philipslumileds.com (110422-VII)

Motor driver kit targets low-power brushless motors

Texas Instruments has launched an evaluation kit for driving three-phase brushless DC (BLDC) and permanent magnet synchronous motors (PMSM). The DRV8312-C2-KIT a high-performance, power-efficient, cost-effective platform is for sensorless field-oriented control (FOC) and sensored or sensorless trapezoidal commutation. It is aimed at brushless motors with operating specs below 50 V and 7 A, used to drive medical pumps, gates, lifts and small pumps, and in industrial and consumer robotics and automation.

The evaluation kit is a ready-to-use motor control and driver solution and includes TI's DRV8312 motor driver, a 32-bit C2000 Piccolo microcontroller (MCU) controlCARD module, a quick-start graphical user interface, full development source code, the Code Composer Studio (CCStudio) integrated development environment (IDE) and a three-phase BLDC motor. It supports sensorless FOC and sensorless or sensored trapezoidal commutation, and it can support sensored FOC with the addition of a separate shaft encoder.

Motor driver evaluation platform for three-phase brushless motors
DRV8312-C2-KIT
Texas Instruments

The DRV8312 three-phase, fractional-horsepower motor driver delivers up to 6.5 A without an external heat sink. It is robust, reliable and features cycle-by-cycle overcurrent, overtemperature, cross-conduction and undervoltage protection. The 32-bit C2000 Piccolo MCU performs control, communication and debug functions. It integrates advanced analogue feedback, digital control peripherals and CPU capability, including access to motor control software modules, real-time debug capabilities, and open-tooled reference designs via free controlSUITE software.

The kit includes an NEMA17 BLDC/PMSM 55-W motor, a 24-V power supply module (with adapters), a DRV8312 baseboard with controlCARD slot, a Piccolo Isolated F28035 controlCARD, USB cable, USB stick with GUI, CCStudio IDE, Quick Start Guide, and link to controlSUITE for all documentation.

The DRV8312-C2-KIT is priced at USD 299, including CCStudio IDE with no compiler or memory limitations. All documentation, software source, and the hardware development package – including bill of materials, schematics and Gerber files – are freely available through controlSUITE.

www.ti.com/drv8312-c2-kit-pr (110422-VIII)

Motion sensor software targets smart consumer devices

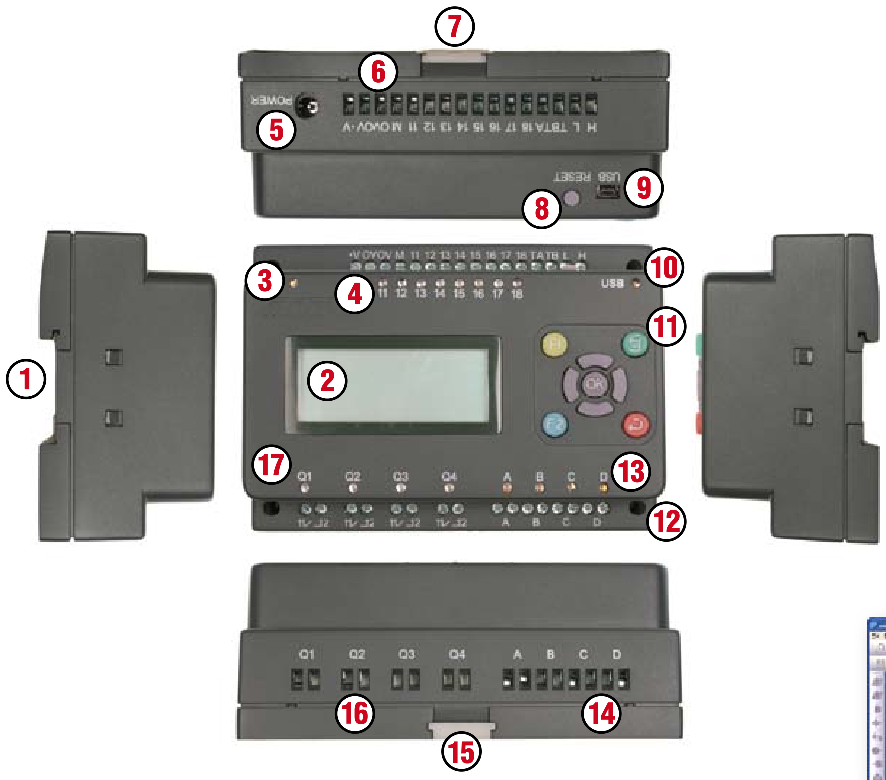
STMicroelectronics has unveiled initial details on its new iNEMO Engine, which incorporates advanced filtering and predictive software to integrate the outputs of a three-axis accelerometer, three-axis gyroscope and three-axis magnetometer. By merging the data from these sensors through sophisticated algorithms, the iNEMO provides substantially accuracy and reliable motion information, which is needed by the makers of next-generation smart consumer devices for a host of new motion-based applications.

Although a single MEMS accelerometer is adequate for many current applications, such as freefall detection, screen rotation and pedometers, a new class of advanced applications that need more sophisticate data is emerging for location-based services, enhanced motion-based gaming, pedestrian dead-reckoning for indoor and multi-floor navigation, robotics and human-body tracking. According to STMicroelectronics, these applications require multiple MEMS sensors, together with advanced software, to achieve better overall system performance in terms of accuracy, resolution, stability and response time.

The iNEMO Engine software integrates a set of highly sophisticated adaptive algorithms for prediction and filtering. The software takes data from the outputs of various motion sensors and integrates it so that the sensor outputs augment each other, surpassing what individual sensors can do alone. The software can correct for magnetic distortions registered by the magnetometer, dynamic distortion measured by the accelerometer, and inherent drift of the gyroscope. This protects the accuracy of heading information, avoids pointing errors and drift problems, and virtually eliminates timeouts for calibration.

www.st.com (110271-VI)

MIAC – the rugged PIC

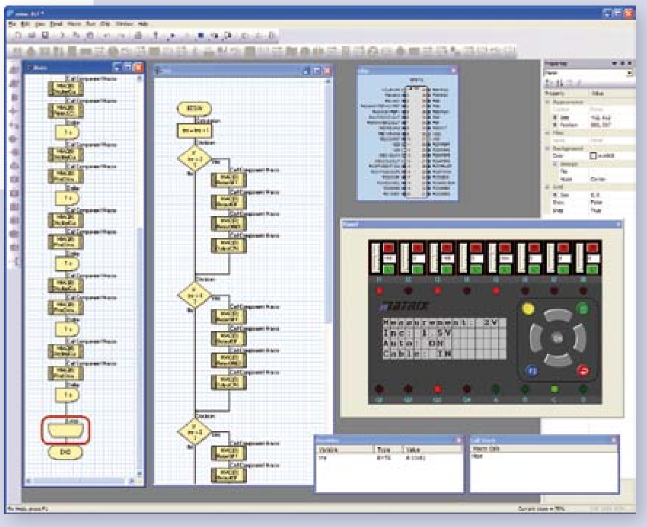


What does it do?

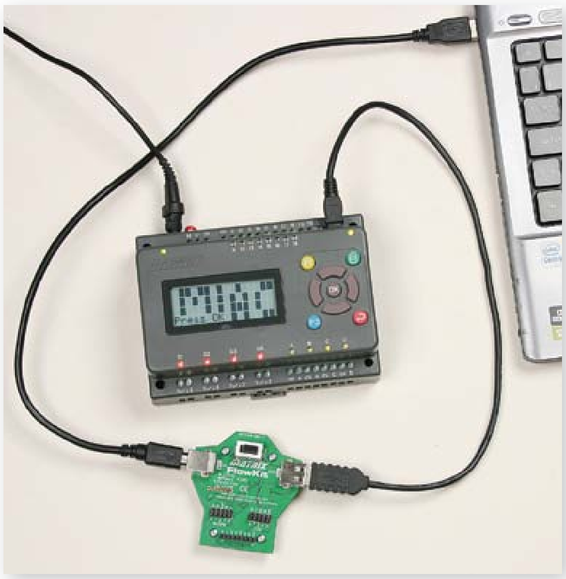
MIAC is an industrial grade control unit which can be used to control a wide range of different electronic systems. It has a lots of applications in industrial control and automation and is perfect for hobbyist PIC projects that need a little oomph.

Key

1. Top hat rail mounting recess
2. 16 character x 4 line LCD display
3. Power LED
4. Input status LEDs
5. 2.1mm power jack
6. Screw terminal inputs
7. Top hat rail retainer clip (upper)
8. Reset / run switch
9. USB socket
10. USB transfer LED
11. Control keys
12. M3 mounting holes
13. Motor status LEDs
14. Motor output screw terminals
15. Top hat rail retainer clip (lower)
16. Relay output screw terminals
17. Relay output status LEDs



***Flowcode** – the graphical programming language supplied with MIAC*



Using MIAC with FlowKit to give full IN-Circuit Debug with Flowcode

Benefits

- Flexible and expandable
- Easy to program with flowcharts, C or Assembly code
- Physically and electrically rugged

Features

- Programmable from USB
- Based on PIC18F4455
- Shipped with a free copy of Flowcode (worth $192.00)
- Compatible with third party C compilers
- 8 digital or analogue inputs
- 4 relay outputs @10amps
- 4 motor outputs @500mA with speed control
- 4 line LCD display and control keys

Create your own PIC projects with the advantageous MIAC Bundle. This package consists of a MIAC Module (in enclosure) and the graphical programming language 'Flowcode 3 for PIC' (Professional-Version)!

Bundle Price:
Only $248.40

Order now at
www.elektor.com/miac-bundle

NEW

ELEKTOR OSPV[1]
An open source vehicle project

The Elektor Wheelie, which we launched a while ago, has been adopted by schools and private individuals who want to gain a thorough understanding of the technology behind self-balancing vehicles.

Our new Elektor OSPV[1] is based on the same concept, but with the difference that it's for indoors, it's easy to steer, it's light and foldable, it's open source, and it looks smart.

It's for indoors

Ultimately it's up to you (subject to local regulations) to decide where you want to travel with the OSPV[1], but it is intended to be used indoors. It's not especially well suited to negotiating bumps or uneven surfaces, since it has a ground clearance of just two centimeters.

The two wheels, which have a diameter of 5.5 inch, are far enough apart to keep you well balanced, but they do not provide much clearance for driving over thresholds or other types of bump.

However, it drives like a charm on any sort of flat floor – including factory floors, school corridors, and the corridors of our office building. The rider stands with both feet outside the wheels. Stepping onto the OSPV[1] takes a bit of practice, but thanks to the low center of gravity it feels very stable.

It's easy to steer

Once you have your feet positioned properly, all you have to do to get the OSPV[1] rolling is to lean forwards or backwards. With the Elektor Wheelie you move the steering column to the left or the right to negotiate curves, but with the OSPV[1] you use a small joystick for this. The vehicle responds immediately to joystick movements, so it's easy to perform pirouettes and other elegant maneuvers. In fact, it would be right at home in a ballroom. The attitude of the OSPV[1] is saved when it is switched on and is subsequently used as a reference, so you should ensure that the steps are level (horizontal) during this process.

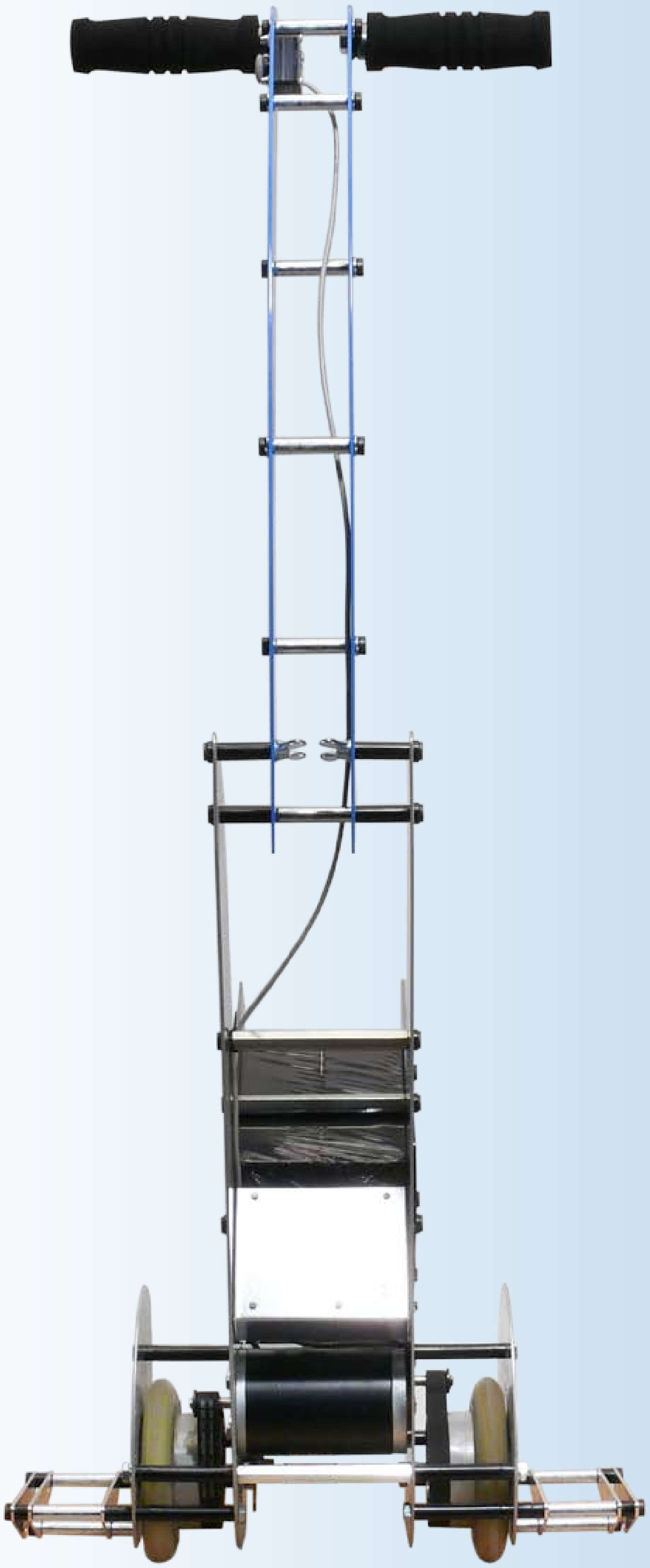
It's light and foldable

The OSPV[1] is designed to be taken apart easily, so it can be stowed conveniently in the boot of a car. It's also relatively light. It weighs just 55 lbs, so you don't have to be a muscleman to pick it up. The pivot and fixing point is on the steering column, which comes apart after you loosen two wing nuts. I'm willing to bet that many of you will be inspired to apply your creativity to this and perhaps come up with a smarter and easier solution. You're welcome to do so – the OSPV[1] is very amenable to simple modifications.

OSPV[1] specifications

Weight:	25 kg (55 lbs)	Ground clearance (under steps):	20 mm (0.8 inch)
Height:	120 cm (47.2 inch)	Foot height (on steps):	5.6 cm (2.2 inch)
Width:	47 cm (18.5 inch)	Width between steps:	29.5 cm (11.6 inch)
Depth:	47 cm (18.5 inch)	Range:	8 km (5 miles)
Maximum load:	90 kg (200 lbs)	Turning circle:	0 m

We're all looking for balance in our lives. With a bit of modern electronics, that's actually not so difficult. Last year we launched the Elektor Wheelie, a self-balancing personal transport device. In this issue we take you for a spin with the new Open Source People Vehicle 1 (OSPV1).



It's open source

The nice thing about the OSPV1 is that you can configure or modify it to suit your wishes. The electronics are the same as in the Elektor Wheelie, the software is freely available, and you can even modify the software to alter the handling characteristics. Everyone who has been involved in developing extensions for the Elektor Wheelie can also use them with the OSPV1. The schematic diagrams, PCB design and source code listing are all available on the Elektor website. The OSPV1 is supplied with a charger for the two 12-V rechargeable batteries, which have a capacity of 9 Ah.

It looks smart

Of course tastes vary, but up to now the OSPV1 has drawn only positive responses. During our test runs we heard comments like "hey, that's neat", "nice device", or simply oohs and aahs. The structure is open and intentionally light, with clean lines.

What you can do with it

The OSPV1 is primarily intended for moving people, but it doesn't have to be limited to that. A variety of other uses are conceivable, ranging from an electric wheelbarrow to a handy motorized shopping cart. This is where the advantages of the open source approach come to the fore. Everyone is free to work on the design and turn it into something special. As already mentioned, it is intended for indoor use, which also includes factory buildings, terminals, school corridors, and so on – as long as the floor is flat, the OSPV1 can run on it.

As for street use, we should point out that using vehicles such as this on the street is subject to rules and regulations, which may vary from country to country. There's also an upper limit on the rider's weight; anything up to 185 lbs is perfectly OK. Finally, you should give some thought to body protection – after all, you're riding on a moving vehicle.

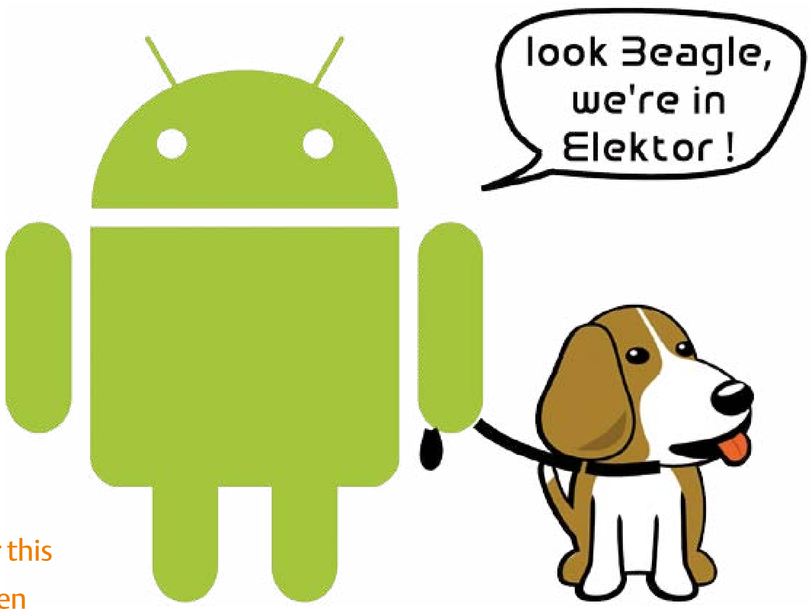
Where to get it

The OSPV1 is exclusively available from the Elektor Shop.
Visit www.elektor.com/ospv1 for full information.

Max. speed:	15 km/h (9.3 mph)	Batteries:	2x lead-acid gel-cell CTM ct0-12L, 9 Ah 12 V
Wheels:	Polyurethane, 14 cm dia. (5.5 inch)	Charger:	Suitable for Europe (230 V) and North America (110 V)
Motors:	DC, 2 x 250 W		
Drive train:	HDT toothed belt	Charging time:	2.5 hours

Developing Apps for Android

Using a PC, BeagleBoard, phone or tablet



By Clemens Valens (Elektor Editorial France)

The first mobile telephone with the Android operating system made its debut in late 2008, and according to current estimates 350,000 new Android phones go on the air each year. What is the reason for this amazing success? Is it Google? Or the fact that it's open source? Or because it works well? In any case, the specific reason isn't that important; what matters is that you can also join the trend and develop your own Android apps.

What is Android?

Android Inc. was founded in Palo Alto, California (USA) in late 2003, for the purpose of developing software for mobile telephones. Scarcely two years later the company was bought by Google. Four years after the founding of Android Inc., in late 2007, the Open Handset Alliance was founded, with Google as a member. The mission of this alliance is to develop open standards for mobile devices. Their first product was Android, a platform for mobile devices that is based on the Linux 2.6 kernel. A year later the first mobile telephone with this new operating system, the HTC Dream, went on the market.

Android is not Linux

Although Android was originally based on Linux, it has now become a fully independent operating system (OS). The source code of Android is still open, but it no longer forms part of the codebase of Linux. Google has modified certain parts of Linux, with the result that Android does not include X Windows or any of the standard GNU libraries. This makes the conversion of existing Linux applications rather complicated. Google also added their own functions, especially with regard to the security of mobile devices.
Android also differs from Linux in terms of the licensing model. Linux is distributed under the GNU General Public Licence (GPL), while Android is distributed under the Apache licence of the Apache Software Foundation (ASF). Unlike the GPL, the Apache licence allows proprietary software that is based on open software (and therefore falls under the same licence) to be distributed without revealing the source code.

Android is also not Java

Although applications ('apps') for Android are written in Java, they are not executed as Java applications because Android does not have a Java virtual engine or Java libraries. This means that Android cannot run Java programs. Android applications use only the syntax of the Java language, and they are executed by a virtual machine called Dalvik.
Incidentally, it is not difficult to develop libraries for Android using other programming languages, such as C or C++. Dalvik is able to import and use libraries of this sort, as well as all Windows dynamic linked libraries (DLLs).

Making apps for Android

Everyone can make apps for Android. All of the tools you need for this are available for download free of charge, and you don't even need to have access to a real Android platform. For instance, on the Android developers website [1] you can find everything you need, including emulators for Android hardware. Software development kits (SDKs) are available for Windows, Linux and Mac OS X environments.
Although Android uses only the syntax of Java, the SDKs are based entirely on Java. This means that you must install the Java development kit (JDK) and you must have a Java virtual machine (a Java runtime environment, or JRE) on your system. Eclipse is recommended as the integrated development environment (IDE), supplemented by the Android Development Toolkit (ADT).
The procedure for installing the SDK (including the JDK, Eclipse, ADT and so on) is described in detail on the relevant website, so there is no need to repeat it here. However, before getting started you should know that the installation process – actually, the downloading process – may take several hours, depending on the speed of your Internet connection, because we're talking about several gigabytes here.
Once you have everything installed, it's a good idea to work through the 'Hello Android' tutorial. This tutorial also provides good explanations, although it doesn't give much guidance on choosing an Android ver-

sion. Several versions of Android are available, with names that reveal a taste for sweet treats: version 1.5 (Cupcake), version 1.6 (Donut), version 2.0/2.1 (Éclair), version 2.2 (Froyo – a brand of frozen yoghurt), version 2.3 (Gingerbread), and version 3.0 (Honeycomb). The hardware that your app can run on depends on the version of Android that you use. Version 3 is intended for tablet computers, while version 2.2 (Froyo) is currently the most popular (and apparently it is even possible to install this version on an iPhone or iPod), followed by version 2.1 (Éclair). You should choose the version that matches your hardware.
Next you need to create a virtual Android peripheral device (Android Virtual Device, or AVD) that is compatible with the Android version you selected, and then launch it. Launching the AVD can take a while, depending on your development machine, and the AVD needs a lot of memory. The higher the Android version, the slower the AVD. On our test PC (Windows XP SP3, 2 GB RAM, Pentium T4200 clocked at 2 GHz), it took several minutes to launch a virtual tablet for use with Android version 3. You just have to be patient.

Android programming can be fun

If you don't have any experience with programming but you would like to generate an app for your Android phone or tablet, there's a tool that's just right for you: App Inventor for Android [2].
To use this tool, you need a PC with a recent operating system (Windows, Linux or Mac OS), an Internet connection, a web browser and Java. To avoid unnecessary disappointments, first visit the App Inventor website to see whether your computer meets all of the requirements. If everything checks out, download App Inventor Setup (nearly 100 MB) and install it with the recommended standard settings.
The next step is to connect your Android device. Things get a bit more complicated here, since you need a driver for the device and this means searching for a suitable driver on the Web, unless you have a phone model that is recognised automatically (such as the Nexus One).
If you (like the author) don't have an Android telephone, you can still start working on your application because App Inventor also includes an emulator. According to Google, launching the emulator can take a few minutes, but on our test machine the emulator was ready to use after approximately 30 seconds.
The procedure for generating an application is the same for both real and virtual Android devices. First you open App Inventor by clicking My Projects on the start page (here we assume that you are logged in via your Google account). This opens the Designer window (**Figure 1**), which is where you develop the visible part of the application by placing objects such as buttons, images and sounds on a background that corresponds to the screen of your Android device. You can adapt the appearance of the application as desired by modifying specific attributes of these objects. The Blocks Editor is where you specify what the application does. Your program is shown in the Blocks Editor as a set of building blocks, with the object attributes and actions taking the form of puzzle pieces. You create a program by linking these pieces together in a particular manner (**Figure 2**). This approach is very similar to Scratch [3], a programming

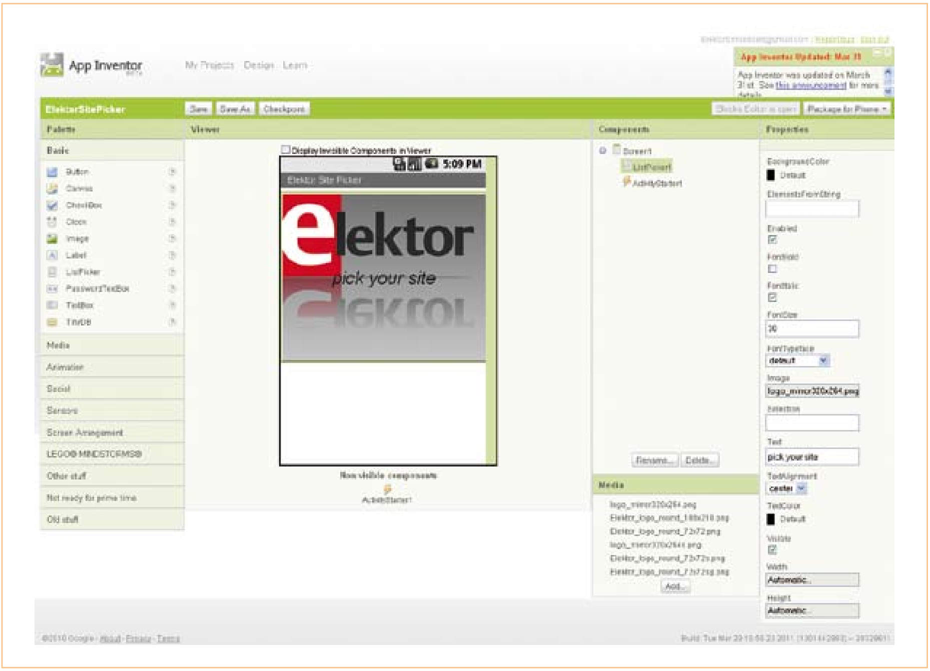


Figure 1. The Designer window of Android App Inventor.

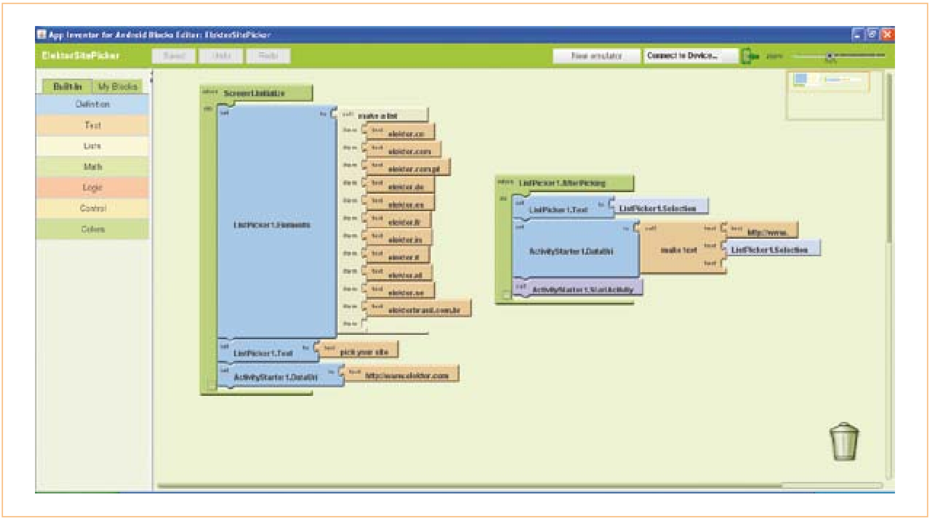


Figure 2. Our test app in the Blocks Editor window.

Caution: pitfalls

The author encountered several problems while installing Android on the BeagleBoard. The first problem occurred right at the beginning of installation script **mkmmc-android.sh**. An error message appeared and only the first partition was created on the SC card. In a sudden flash of insight, the author realised that the *grep* instruction in the script was looking for the word ‘disk’, but he had a French version of Ubuntu and this meant that the instruction should have been looking for the word ‘disque’. Instead of modifying the script, the author changed the language setting of Ubuntu in order to avoid similar problems with other scripts. If you do change the script and then save it with a different name, remember to also generate an executable version (right-click and select *properties -> permissions*).

The second problem involved video output. The SD card included with the BeagleBoard holds a copy of the Ångström distribution, which is a version of Linux adapted to embedded systems. When the BeagleBoard was started up with this operating system, everything worked fine and an image appeared on the monitor. By contrast, attempts to start up with Android or Ubuntu resulted in a black screen, despite the fact that the monitor worked perfectly with the computer. The cause of this problem turned out to be the video resolution, which was not compatible with the resolution of the monitor. The author discovered that the Android and Ubuntu distributions for the BeagleBoard are configured with a default resolution of 1280 □ 720 pixels, which is somewhat unusual and which the monitor concerned could not handle.

To solve this problem, you can edit the **boot.scr** file by using the **mkbootscr** tool, which is located in the *Tools* folder of the TI SDK, but this script produces a boot.scr file for a different card. Fortunately it also generates a boot.cmd file, which can be modified with a text editor and then converted into a boot.scr file.

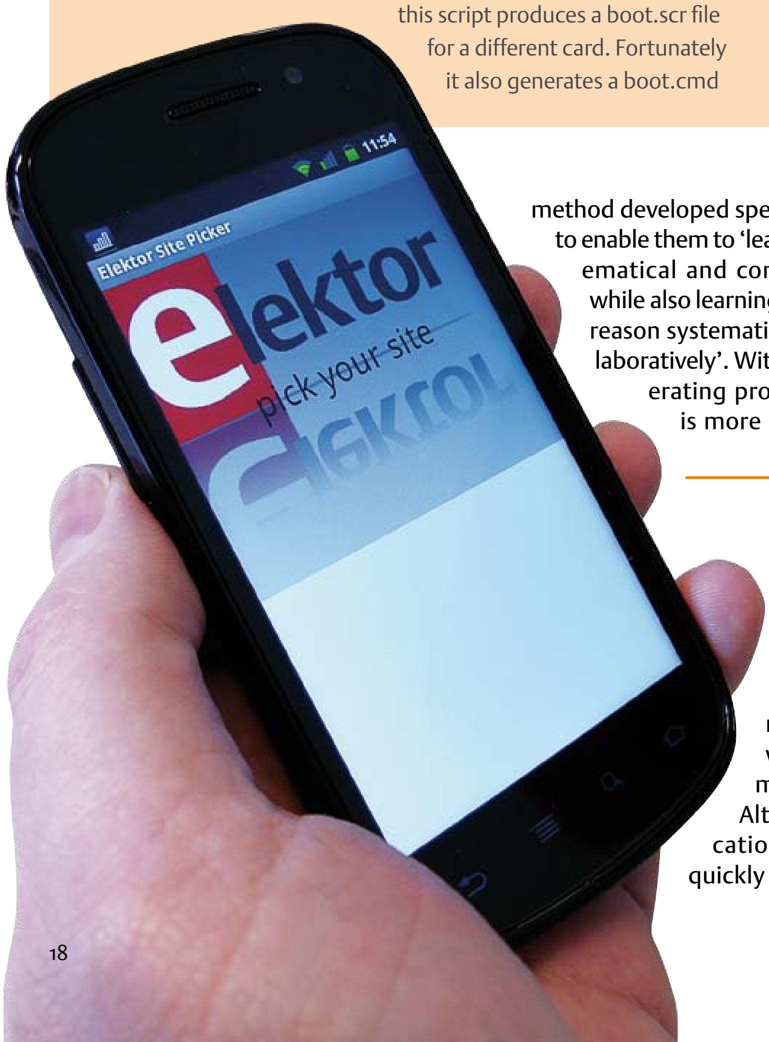
In the boot.cmd file, replace the **bootargs** by the corresponding values in the user guide for the TI SDK and add the following line at the end (before the ‘´ ’ character):

omapfb.mode=dvi:1024x768MR-16@60

If necessary, you can replace ‘1024x768’ by a different resolution compatible with your monitor. Now issue the command **mkimage -A arm -O linux -T script -C none -a 0 -e 0 -n ‘Execute uImage.bin’ -d boot.cmd boot.scr** to create the boot.scr file and copy it to the SD card in the root directory of the boot partition.

The TI application note specifies a clock frequency of 1 GHz for the boot.scr file (**mpurate=1000**), but there are reports all over the Web saying that the BeagleBoard does not work well at especially high clock rates. Consequently, we did not try this frequency and chose a lower frequency instead (800 MHz) by setting **mpurate=800**. The author’s boot.scr file (which works) is available at [4].

Another pitfall that you should avoid relates to user permissions for manual initialisation of the SD card (which is certainly possible). This must always be done by someone acting as a root user, as otherwise Android will not be able to complete the startup process and will send messages containing the phrase ‘Permission denied’ to a (virtual) terminal connected to the serial port of the BeagleBoard. In order to act as a root user of Ubuntu, open a terminal window and type **sudo -s**. Now you can manually initialise the SC card from this terminal window. Note that the boot partition of the SC card must be a bootable partition. Also remember to execute the command **/media/rootfs# chmod –R 755 /mnt** in order to grant access to the *rootfs* file system.



It really works: our test application on a genuine Android mobile phone.

method developed specifically for children to enable them to ‘learn important mathematical and computational ideas, while also learning to think creatively, reason systematically, and work collaboratively’. With this method, generating programs for Android is more like fun than work, even for Elektor readers (and others) with no programming experience. Although small applications can be created quickly using this method, some things are not possible and it’s not always simple. Furthermore, App Inventor is a beta version, which means it may still contain bugs. For example, while experimenting with this tool the author had major problems with the voice synthesizer when the application also included an ActivityStarter object.

Once your application is ready, you can package it (*Package for Phone*) and download it to your PC or directly to your Android device. If your phone is not connected to the PC, you can obtain a flash code (QR code; see **Figure 3**) with a link to the application, so that it can be downloaded to the phone over the air (assuming that the phone can read flash codes and can connect to the Internet). You can also retrieve the source code from the project management page. For example, you can download the author's designs from [4] and import them into your project.

Required hardware

As an electronics enthusiast, you may be interested in building your own hardware platform for Android, but first you need to know the specifications. Unfortunately, they are a bit vague, but it appears that the absolute minimum is an ARM microcontroller clocked at 200 MHz with 32 MB of RAM and 32 MB of flash memory. However, it is recommended to have at least 128 MB of RAM and 256 MB of flash memory. For Android 3 you need a microcontroller clocked at 1 GHz, along with 512 MB of RAM. With regard to peripherals, you need a USB port, a (flatscreen) QVGA monitor with 65,536 colours or more, and ten buttons or keys. Other useful (but not essential) items are an SD or microSD card reader, a camera with 2-megapixel resolution, and a Bluetooth interface.

Of course, you can put together a suitable system on your own, but it is much easier and probably cheaper to buy a ready-made board. With a bit of searching on the Web you're bound to find something that fits your budget.

BeagleBoard

A few years ago Texas Instruments (TI) developed a board called the BeagleBoard [5], which is intended to promote interest in their OMAP family of multimedia processors. This board contains open hardware and is supported by the open source community. Although the schematic diagrams and PCB artwork for the board are available for free, you're out of luck if you want to build your own BeagleBoard because the RAM is mounted on top of the processor using the Package on Package (PoP) technique. Fortunately, the board is not especially expensive (the xM version costs only USD 149), and it is readily available.

The BeagleBoard has an extensive set of features. The most recent version (xM), shown in **Figure 4**, includes four USB ports (host and peripheral), an Ethernet port, an RS232 port, a USB OTG port, a microSD connector, stereo audio input and output, an S Video output, an HDMI port and expansion connectors for an LCD screen, a camera, and a breadboard area. The processor, or more precisely the system on chip (SoC) device, is a DaVinci DM3730 digital media processor, which integrates an ARM Cortex-A8 core, a TMS320C64 DSP and an SGX graphics accelerator. The processor is compatible with high-performance operating systems such as Windows CE, Linux, QNX and Android.

Unlike the first version of the BeagleBoard, the xM version does not have any flash memory. In this version the RAM is complemented by a microSD card, which holds the operating system and the applications. The advantage of this configuration is that it is easy to change operating systems by replacing the SD card. The drawback is that performance is reduced slightly, since SD cards are not noted for their speed. In some cases the performance can be improved by using a USB stick.

Barcode link for ElektorSitePicker

OK

Note: this barcode will only work for user ElektorEmbedded@gmail.com. See the FAQ for info on how to share your app with others.

Figure 3. The flash code that gives a mobile phone access to our application.

All in all, the BeagleBoard is a powerful and versatile computer, which can be used for a wide variety of applications. Although it is not necessarily the best platform for Android, it is a reasonable alternative if you don't have an Android phone or an Android tablet.

For this development platform you also need:

- an external 5 V power source that can accommodate a 1 A load (we measured a power consumption of nearly 3.5 W

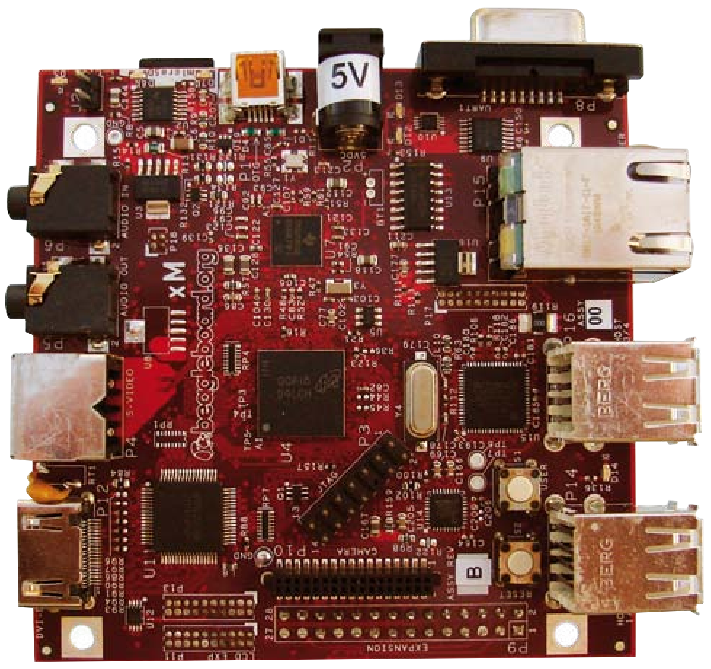


Figure 4. The BeagleBoard xM operates from a 5 V supply and draws nearly 700 mA when Android is running.

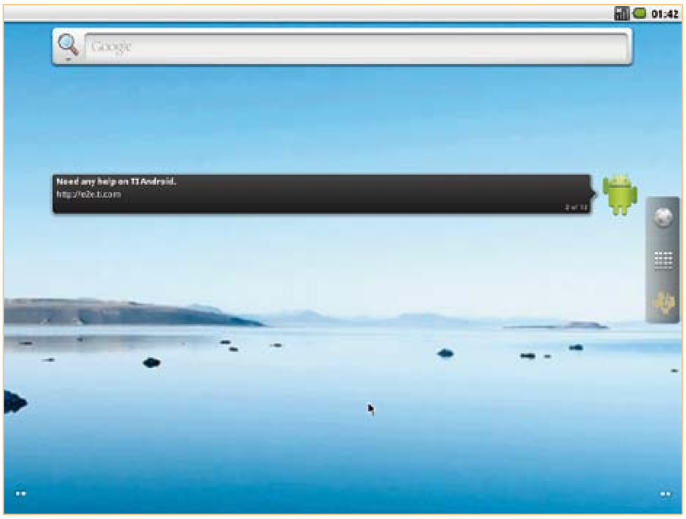
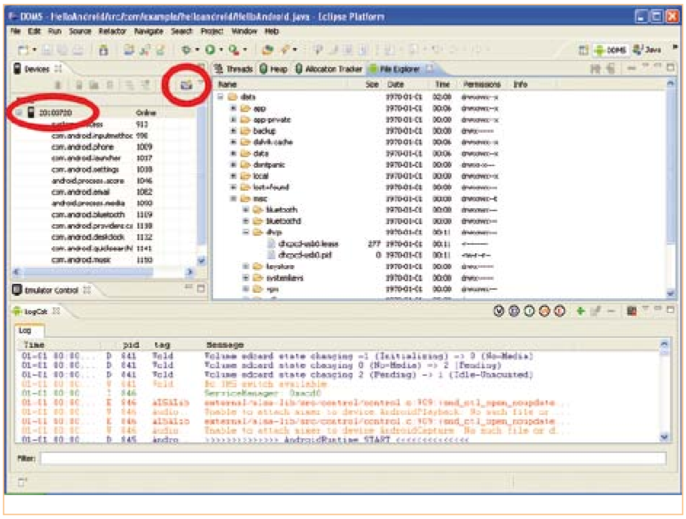


Figure 5. The Android start screen on the BeagleBoard monitor.



DDMS - HelloAndroid/src/com/example/helloandroid/HelloAndroid.java - Eclipse Platform
File Edit Run Source Refactor Navigate Search Project Window Help
Devices
Threads Heap Allocation Tracker File Explorer
Emulator Control
LogCat
Log
Filter:

```
20100720                         Online
  com.android.inputmethod        998
  com.android.phone              1009
  com.android.launcher           1017
  com.android.settings           1038
  android.process.acore          1046
  com.android.email              1082
  android.process.media          1090
  com.android.bluetooth          1119
  com.android.providers.ca       1138
  com.android.deskclock          1132
  com.android.quicksearchl       1141
  com.android.music              1150
```

```
Time           pid  tag       Message
01-01 00:00... D 841  Vold      Volume sdcard state changing -1 (Initializing) -> 0 (No-Media)
01-01 00:00... D 841  Vold      Volume sdcard state changing 0 (No-Media) -> 2 (Pending)
01-01 00:00... D 841  Vold      Volume sdcard state changing 2 (Pending) -> 1 (Idle-Unmounted)
01-01 00:00... W 841  Vold      No UMS switch available
01-01 00:00... I 846            ServiceManager: 0xacd0
01-01 00:00... E 846  ALSALib   external/alsa-lib/src/control/control.c:909:(snd_ctl_open_noupdate ...
01-01 00:00... W 846  audio     Unable to attach mixer to device AndroidPlayback: No such file or ...
01-01 00:00... E 846  ALSALib   external/alsa-lib/src/control/control.c:909:(snd_ctl_open_noupdate ...
01-01 00:00... W 846  audio     Unable to attach mixer to device AndroidCapture: No such file or d...
01-01 00:00... D 845  andro     >>>>>>>>>>>>>> AndroidRuntime START <<<<<<<<<<<<<<
```

Figure 6. The DDMS window in Eclipse.
Our Android device is called '20100720'. You can use the camera icon to make pictures of the screen.

with Android running, equivalent to 700 mA at 5 V);

- a monitor with a DVI-D port (although the BeagleBoard has an HDMI port, it outputs only DVI-D signals);
- a USB hub with an external power source (because the board's USB ports cannot supply enough current);
- a USB keyboard and a USB mouse, both connected to the hub.

A microSD card with a capacity of at least 4 GB is recommended for installing Android (or Ubuntu). A suitable card is included with the BeagleBoard, but the author purchased an additional 8 GB microSD card in order to avoid damaging the Ångström distribution already present on the supplied card (Ångström is a Linux distribution for embedded systems).

Although TI provides everything you need to start using Android, you must have access to a Linux PC (or a virtual Linux PC) with an SD card reader in order to initialise the SD card. TI recommends Ubuntu 8.04 or later; we used Ubuntu 10.4. After downloading TI_Android_FroYo_DevKit-V2.2 (1.1 GB) [6], you're ready to get started. Although the procedure for this is essentially very simple, you may encounter a few problems. Consult the 'Caution: pitfalls' inset if anything goes wrong.

Open a terminal window on the Ubuntu machine (for example, press Alt F2 and tick '*Run in terminal*'), then enter **sudo -s** in order to become a root user with all of the associated permissions. Although the procedure described below is not absolutely necessary, since the card can also be initialised using **sudo** alone, it is much easier to use if you encounter any problems.

Unpack the archive (**tar -xzvf TI_Android_FroYo_DevKit-V2.2.tar.gz**) and use the **cd** command to change to the directory (folder) **TI_Android_Froyo_DevKit-V2.2\Prebuilt_Images\beagleboard-xm\**, which is now present on your computer. In this folder, use the command **./mkmmc-android.sh /dev/sd<device>** (preceded by **sudo** if you are not a root user) to launch the script **mkmmc-android**. In this command, **<device>** must be replaced by the drive letter of the card reader. On the author's system this is 'C', so the command becomes **/dev/sdc**. There are various ways to find out which drive letter is assigned to the card reader; we used the Ubuntu disk utility (under *System -> Administration*). Be sure to get this right, since the Android installation script formats the drive specified by the command.

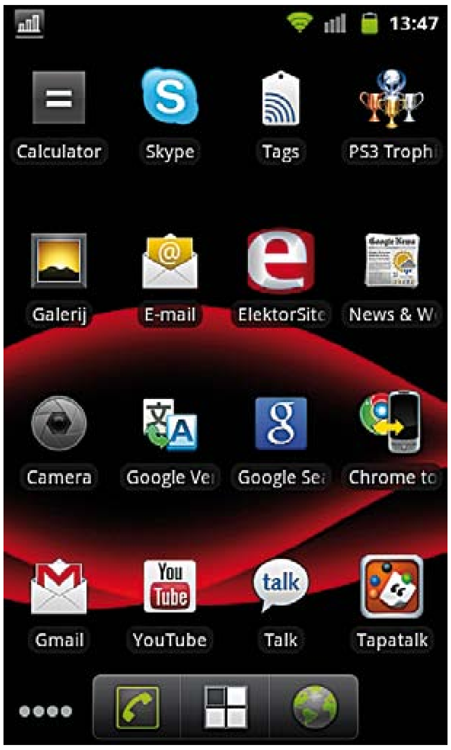
It may take a while for this script to complete execution, but once it's finished without any problems you are done with Ubuntu. The formatted SD card now has three partitions: *boot*, *rootfs* and *data* (with sizes of 74 MB, 4 GB and 3.9 GB, respectively, on an 8 GB SD card). Plug the SD card into the socket on the BeagleBoard and start it up. This takes fairly long the first time (more than a minute in the author's case), but in the end you will see the word 'Android' appear on the monitor, followed by the Android start screen (**Figure 5**). It is instructive to connect a (virtual) terminal to the RS232 port of the BeagleBoard (port settings: 115200, N, 8, 1) and view the messages exchanged during the startup process.

Bear in mind that Android is intended to be used with touchscreens and the entire user interface (HMI) is based on motions (touching and dragging). If you do not have a touch-sensitive screen, you can use the mouse as a finger. You 'touch' an object by left-clicking it, and you drag an object by clicking and holding the left mouse button. A right-click means 'Go back'. You can do almost everything with the mouse, except opening the menu (at least we weren't able to do so). This is not much of a problem, unless you want to change the background image for the screen (we were unable to find an alternative method for this). In order to open the menu, you must press the F1 key on the keyboard. Incidentally, this was the only key that we used during all of our Android sessions.

As soon as the BeagleBoard was able to launch Android without any problems, we said goodbye to Ubuntu and continued with Windows XP for the sake of simplicity. This choice was also motivated by a shortage of monitors. The only flatscreen monitor we had available had to be used for both Ubuntu and Android, which ultimately became a bit awkward, and Windows XP was running on a laptop with its own screen.

The next step is to integrate the BeagleBoard into the Android SDK environment. (If you haven't already installed this SDK, see the section 'Making apps for Android' above for more information.) How you go about this depends entirely on your operat-

Our icon joins the crowd. Pretty cool!



ing system and is explained very well in the TI application note. Here we describe the procedure for Windows, since this is what we used.
First download the file **usb_driver_r03-windows.zip** [7] and unpack it. Open a terminal window and enter the following command (followed by Enter):

```
echo 0x18D1 > "%USERPROFILE%\.
android\adb_usb.ini"
```

This cryptic command creates a text file named **adb_usb.ini**, which contains the text '0x18D1', in a folder named **.android** located in the folder where Windows saves your personal settings. If you want to see where this folder is located, type **echo %USERPROFILE%**.
Next, open the file **android_winusb.inf** in Notepad. This file is located in the folder containing the USB driver that you unpacked earlier on. In this file, insert the following lines below the header **[Google.NTx86]**:

```
; Beagle Board
%SingleAdbInterface% = USB_Install,
USB\VID_18D1&PID_9018
%CompositeAdbInter-
face%  =  USB_Install,  USB\
VID_18D1&PID_9018&MI_01
```

Save the file and close it.
Start up the BeagleBoard and wait until the Android start screen appears. Connect a USB cable between the PC and the mini-USB connector of the BeagleBoard, which is located next to the power connector, and install the driver under Windows. This must be done manually by telling Windows that the driver is located in a place specified by you. Enter the path to the modified **android_winusb.inf** file and let Windows do the rest of the work.
After Windows has recognised the BeagleBoard as an Android ADB interface (which requires restarting the computer), you can test the link by entering the commands shown in boldface below in a terminal window:

```
adb kill-server
adb start-server
* daemon not running. starting it
now on port 5037 *
* daemon started successfully *
adb devices
List of devices attached
20100720        device
```

The commands and responses listed above indicate that your BeagleBoard has been recognised as an ADB peripheral device with the name '20100720', and from now on you can use it as an Android device in App Inventor or Eclipse. It will log on with this name (number) when you run the application you have developed for it.
You can also use the terminal window to install applications (**adb install my_app.apk**), such as applications you have downloaded from Android Market, or to uninstall applications (**adb uninstall my_app.apk**). Alternatively, you can use application management to uninstall applications directly in Android.
The command **adb shell** opens a terminal window for Android, where you can enter Linux commands to perform operations on files.
You can use the Dalvik Debug Monitor Server (DDMS) perspective in Eclipse for tasks such making pictures of the screen of your Android device by clicking the small camera icon (**Figure 6**). This tool can also be used outside the Eclipse environment. It is located in the *Tools* folder of your SDK installation under the name **ddms.bat** (what else would you expect?).
If you want to connect your BeagleBoard with Android to the Internet, it's worthwhile knowing that the Ethernet port, like the four USB ports on the board, is actually connected to the USB hub integrated on the board. Consequently, the Ethernet port is not designated **eth0**, but instead **usb0**. This means that **eth0** must be replaced by **usb0** in commands related to the network. To obtain an IP address, connect the BeagleBoard to your network and enter the command **netcfg usb0 dhcp** in the ADB shell (assuming that your DHCP server is online). Request the IP address of your DNS (**getprop net.usb0.dns1**), tell Android that it must use this address to access the Internet (**setprop net.dns1 <received IP address>**), and voilà: your BeagleBoard is online.
You now have all the information you need to start developing your own Android apps. We look forward to seeing the first user-developed app that allows your own circuit to be controlled via Bluetooth.

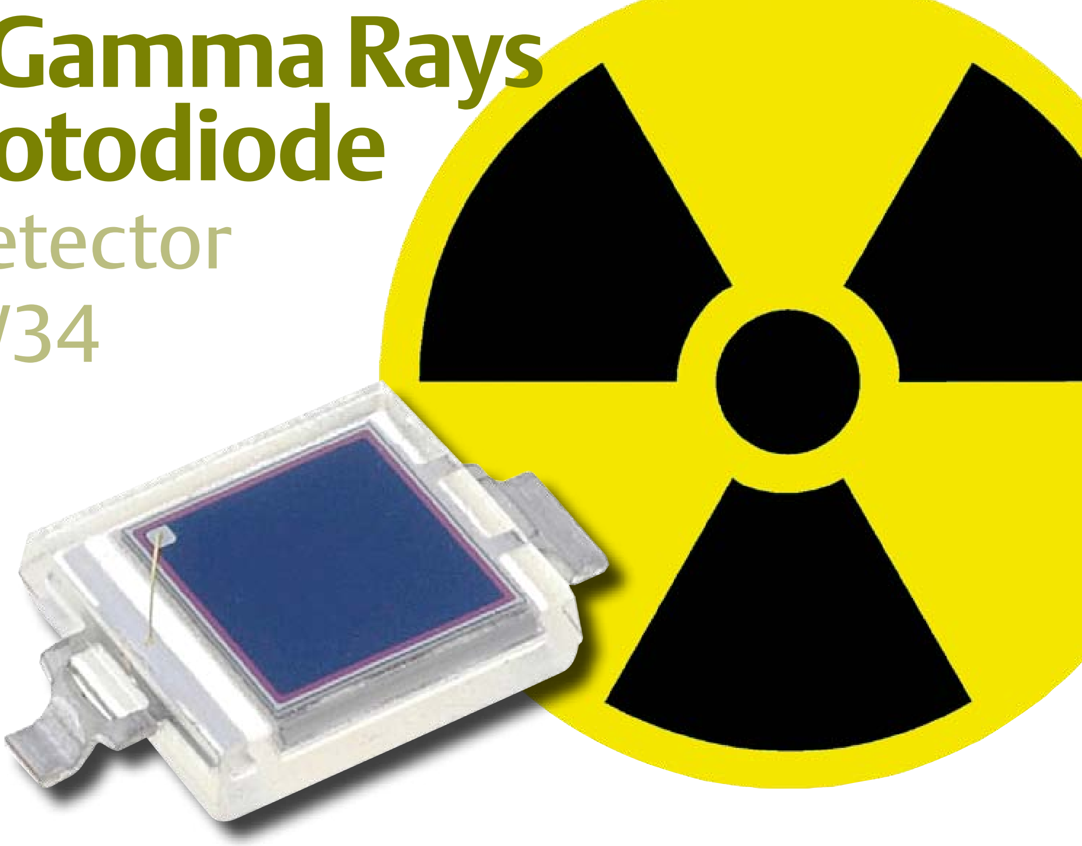
(110265-I)

Internet Links

[1] http://developer.android.com/index.html
[2] http://appinventor.googlelabs.com/about/
[3] http://scratch.mit.edu/
[4] www.elektor.com/110265
[5] http://beagleboard.org
[6] http://software-dl.ti.com/dsps/dsps_public_sw/sdo_tii/TI_Android_DevKit/02_02_00/index_FDS.html
[7] https://dl-ssl.google.com/android/repository/usb_driver_r03-windows.zip

Measure Gamma Rays with a Photodiode

Radiation detector using a BPW34



by Burkhard Kainka (Germany)

The first device that springs to mind when thinking about measuring radioactivity is the Geiger-Müller tube. However, these counter tubes are getting hard to find and expensive, and even if you do manage to get hold of one, you will still need to find a way to generate its operating voltage of several hundred volts. It is less well known that even a humble photodiode such as the BPW34 can be used to detect X-rays and gamma radiation.

Ionizing radiation is potentially harmful to health, and it is important to minimize one's exposure to it as far as possible. A simple Geiger counter with a small glass mantle tube will not usually be adequate to detect possibly harmful radiation. The semiconductor sensor we describe below also has a relatively low sensitivity, only being able to detect fairly intense sources of radiation, but it is nevertheless an interesting device for carrying out experiments and measurements.

An advantage of using a photodiode is its small sensitive area. The background rate due to cosmic rays is very low and signals from small samples are easier to detect than with a counter tube.

Radiation

When considering protection from radiation it is gamma rays that are the most important. They can penetrate walls and it is difficult to block them. Hard gamma rays are present in the environment all around us and are also not stopped even by a thick wall. Alpha particles, on the other hand, only have a short range and generally cannot even penetrate a sheet of paper: this is the reason that many counter tubes cannot detect them, unless they have a very thin mica window. Beta particles have a longer range and can penetrate thin sheets of metal. Most counter tubes are mainly designed for detecting gamma rays while, within certain limitations, also being sensitive to beta particles.

Diode as detector

The behavior of a type BPW34 PIN photodiode is similar to that of a low-cost counter tube. Alpha particles will be stopped by the plastic enclosure of the device, whereas gamma rays pass through without problem and create many electron-hole pairs in the diode's depletion layer. If the diode is reverse-biased, almost all of the charge carriers will be drawn away: this corresponds to a small current pulse which can be amplified and processed. Beta particles can also generate such a signal if they are sufficiently energetic to reach the depletion layer.
The amplitude of the signal produced by the photodiode is considerably smaller than that normally obtained from a counter tube, and so a very low-noise instrumentation amplifier circuit is needed.

Another requirement when using a photodiode as a beta and gamma radiation detector is that light must be completely excluded, as otherwise the photocurrent will overwhelm the signal we are looking for. In our prototype we used ordinary aluminum kitchen foil as a screen.

The difference between PIN diodes and PN diodes is that the former include an extra very lightly N-doped region called the 'intrinsic', or 'i' region. This high-resistance region lies between the 'n' and 'p' regions. The result is a wider depletion layer in the diode, and hence a greater volume of semiconductor that can interact with photons. The structure is used in a photodiode in order to obtain as many charge carriers as possible per photon, optimizing the device's sensitivity.

Another way to increase sensitivity is to increase the sensitive area of the device. However, this has the disadvantage of increasing its capacitance, which reduces the (voltage) amplitude of its output signal. Commercially-available semiconductor radiation detectors have a large area and a wide intrinsic region. Simple PIN photodiodes such as the BPW34 are less sensitive than these devices, but also of course somewhat cheaper.
The BPW34 and BPX61 photodiodes are practically identical apart from their enclosures. The (cheaper) BPW34 comes in a plastic package, whereas the BPX61 comes in a TO-5 metal enclosure with a glass window. It is possible to remove this window (carefully!) to expose the chip: this will make the diode capable of detecting alpha particles.
The rays or particles must first make it through a 15 µm thick piece of aluminum (the thickness of ordinary kitchen foil). This is no obstacle to gamma rays and beta particles, and alpha particles with an energy of 4 MeV or more will also pass through. When the particle enters the plastic of the photodiode package, deceleration radiation (German: 'bremsstrahlung') will be produced in the form of brief flashes of light, which can also sometimes be detected by the sensor. It is therefore not impossible for even the BPW34 to have some sensitivity to alpha particles.

In principle any semiconductor is sensitive to ionizing radiation. It is perhaps less surprising, then, that a photodiode is sensitive to radiation than that the effect has not been widely remarked on before. The effect is however well known in dynamic RAMs, whose stored data can be corrupted by incident radiation. The problem of building electronics to withstand the higher levels of radiation found in space is becoming increasingly difficult, because as structures get smaller it becomes increasingly likely that a single energetic particle can interfere with the operation of a circuit.

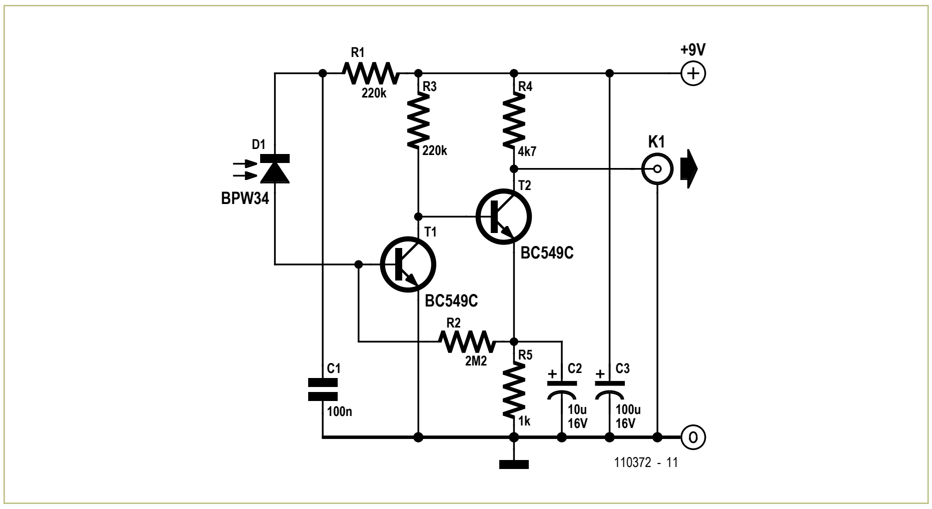


Figure 1. The amplifier circuit.

Amplifier

In the literature charge amplifiers are usually constructed using a low-noise FET-input opamp as the input stage. Here we take an alternative approach: **Figure 1** shows the circuit of the sensor amplifier. Two transistors are used to amplify the signal from the photodiode. The direct-coupled amplifier automatically sets itself to a mid-range operating point, which gets a good signal-to-noise ratio from the low-noise BC549C transistors.
The transistor input of the amplifier has a comparatively low impedance, which gives good noise matching. As a result of its base-collector capacitance the first stage also operates as an integrator: this turns the brief pulses from the photodiode into longer pulses which can then more easily be amplified.
Sensitivity can also be increased by increasing the reverse voltage on the diode. This reduces the capacitance of the diode and increases the size of the depletion layer. The voltage can be as high as 32 V, although the optimum value probably lies somewhat lower: the diode already operates well at 9 V. It is also possible to wire two or more photodiodes in parallel, and that way it is possible to achieve a sensitivity on a par with that of a small counter tube such as the ZP1310.
An oscilloscope can be connected to the output of the circuit to view the signal. Readers who yearn for the clicking sound of a 'real' Geiger counter should consult the text box 'From radiation to sound' for a suitable solution.

Construction

The circuit can be built on a piece of breadboard (see **Figure 2**), with the photodiode on the underside (**Figure 3**). To keep light out of the sensor the whole circuit is wrapped in aluminum foil (**Figure 4**). As mentioned above, ordinary kitchen foil is

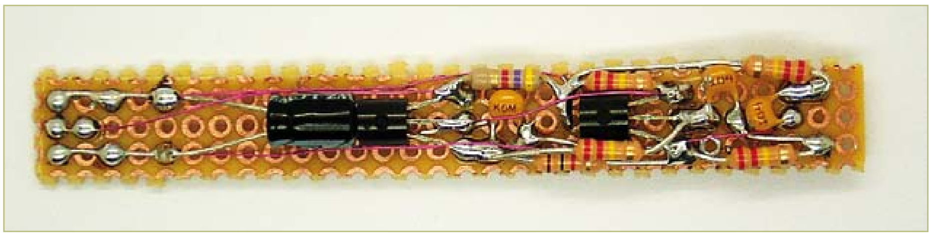


Figure 2. Prototype of the sensor amplifier.

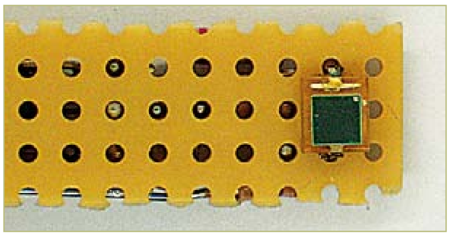


Figure 3. The sensor is on the underside of the board.

Luminous dials

An old watch with a luminous dial is ideal for testing radiation detectors. Alternatively, a suitable alarm clock or compass might be found at a car boot sale.

Radioactive luminous paints were used until about 1965 and watch and clock faces from that time will have lost almost all of their luminosity by now. If you are not sure whether your watch is radioactive, it is possible to carry out a simple test without any electronics: all you need is a magnifying glass. In complete darkness, with your eyes fully accommodated, look at the hands under the glass. If the paint is a radioactive mixture, you will see faint flashing and flickering: you are actually witnessing individual decays. The alpha particles produced excite the luminous paint. If you see no light or an absolutely uniform light, then there is no radioactivity present.

This test is probably only possible with luminous paints based on radioactive materials that have aged considerably, as when the paints are new there may be too many decays happening to see them individually.

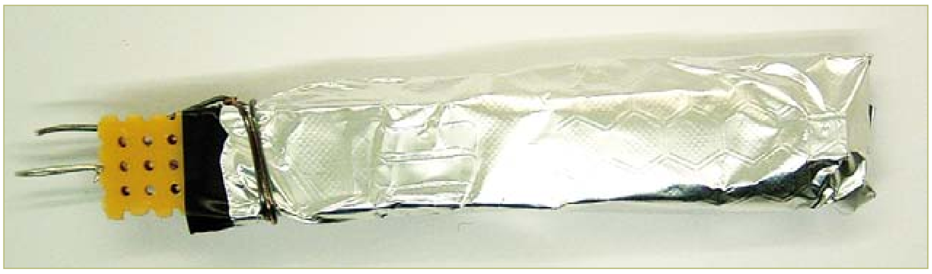


Figure 4. The whole thing is wrapped in aluminum foil.

ideal for this as it is thin enough to let beta particles through. The foil also functions as electrical screening.

To avoid the foil causing short circuits, wrap the board first in insulating tape, leaving a gap for the window of the photodiode. Then wrap the assembly in foil, not forgetting to connect the foil to ground.

Experiments and results

The best way to evaluate the results is to use a digital oscilloscope, in AC-coupled mode. A good place to start is with a vertical sensitivity of 50 mV per division and a timebase of 0.2 ms per division. Some oscilloscopes have a persistence mode which allows the results to be accumulated on the display. It is of course also possible to use an analog oscilloscope.

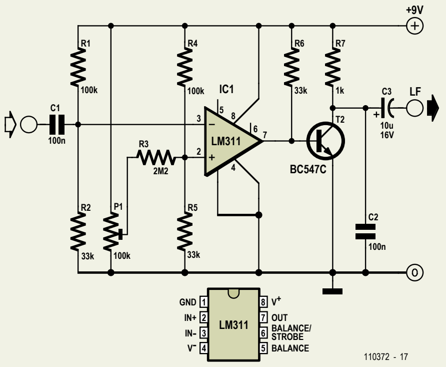
In the quiescent state a band of amplified noise about 30 mV_{pp} will be seen (**Figure 5**). A gamma ray hitting the sensor will be seen as a positive pulse with a small negative undershoot following it. If strong negative-going pulses are seen it is a sign that the circuit is not screened well enough and is reacting to RF signals: the radiation we are trying to detect only causes positive-going pulses. **Figure 6** shows the signal accumulated over a period of 30 s with the sensor pointing at an old pocket watch with luminous hands.

Figure 7 shows measurements taken from another radioactive sample, in this case a small piece of pitchblende (uranite), a naturally-occurring ore of uranium. Again the measurements are taken over 30 s. It is easy to see that this sample is more radioactive; but it is also possible to see a difference in the energy distribution. There are more

From radiation to sound

A 'real' Geiger counter makes a pleasant ticking sound. Our diode sensor, on the other hand, is completely silent.

We can remedy the situation with the help of a comparator and a circuit to stretch the pulses so that we can drive a loudspeaker to make clicks. The tested circuit shown here uses a type LM311 comparator which produces a pulse on its output when the amplitude of a pulse on its input exceeds a threshold set by the trimmer. The transistor at the output stretches the pulse to make it audible. The final output can be used to drive headphones, an audio amplifier and a loudspeaker, or a PC-style active speaker.



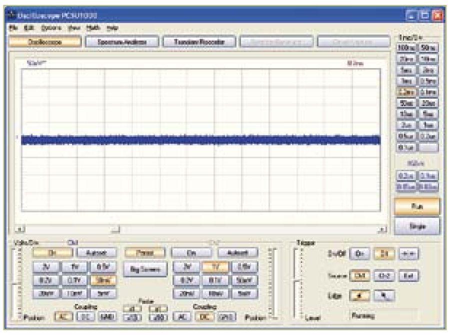


Figure 5. Circuit output in the quiescent state.

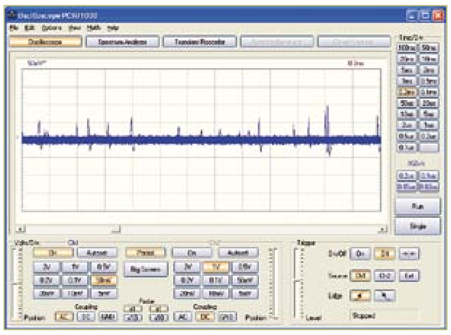


Figure 6. Readings over 30 seconds from an old watch with luminous hands.

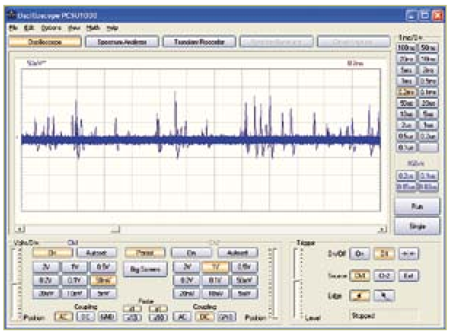


Figure 7. Readings over 30 seconds from a sample of a mineral containing uranium.

pulses with an amplitude of over 100 mV than in the case of the luminous watch. This shows that, unlike a Geiger tube, this detector can determine the energy of the individual particles. This in turn lets us make deductions about the types of nuclei that are disintegrating. In the case of pitchblende these will be elements in the uranium decay series; in the case of the luminous watch the decaying nucleus is likely to be radium.

The possibility of accumulating readings over a long time period lets us examine samples where we would expect little or no radioactivity. Here the photodiode works better than the counter tube as the background rate is practically zero. With a Geiger counter there are almost always pulses arising from cosmic rays: these hard gamma rays also affect the photodiode sensor, but because of its much smaller sensitive area these events are much rarer, and so it is easier to separate the wanted signal from the background. **Figure 8** shows readings from a sample of galena, a mineral that we would expect not to be radioactive at all. After half an hour, however, we see two clear peaks. We obtained a similar result from a sample of granite, which is known to be slightly radioactive.

Certain components and pieces of apparatus manufactured before stricter modern

Advertisement

PROPELLER MULTICORE STARTUP SPECIAL!

$79.99

A great way to play with the multicore Propeller P8X32A microcontroller at a great price! This limited-time-only kit includes a Gadget Gangster Propeller Platform USB, a ProtoPlus module, and a cool pinstriped acrylic enclosure to house your completed project! Combining a 64 KB EEPROM, 5 MHz removable crystal, 1.5 A power regulation, USB, and a microSD card slot on a compact, breadboard and protoboard friendly module, the Propeller Platform USB is an easy-to-use development tool for the multicore Propeller microcontroller. All 32 Propeller I/O are available via pin sockets, along with 5 and 3.3 V regulated power. The ProtoPlus module adds video and audio to your Propeller Platform and includes a prototyping area. Traces in the prototyping area are also marked in the silkscreen, so it's easy to see how the traces are connected. **We have several tutorials available online to get you started.**

PARALLAX INC

www.**parallax**.com

"ParallaxInc" on Twitter, Facebook, and YouTube

Order the **Propeller Multicore Startup Special** (#910-32316; $79.99) at www.parallax.com or call toll-free 888-512-1024 (M-F, 7AM-5PM,PDT).

Friendly microcontrollers, legendary resources.™

Prices subject to change without notice.
Parallax, the Parallax logo, and Propeller are trademarks of Parallax Inc.

Radon decay products

Radioactive samples for testing can be obtained directly from the environment: we are constantly surrounded by radioactive materials. For example, radon is continuously escaping from the ground. This radioactive gas decays (with a moderate half-life) producing further radionuclides, which can be collected.

Take a thin (0.2 mm diameter) piece of enameled copper wire and stretch it out indoors. Apply a negative potential of between –5 kV and –10 kV to the wire and leave it for ten minutes. Disconnect the high voltage and then run a strip of paper along the length of the wire: you will pick up a dark line of dust that was attracted to the wire by the high voltage.

These dust particles are particularly rich in the radioactive decay products of radon. The reason for this is interesting: when the radon decays the new nuclei are moving rapidly, and are therefore stripped of some of their electrons. This means that they have a positive charge and so are attracted to the negatively charged wire.

When this 'dirty' strip of paper is held up to the radiation detector, a high level of activity will be recorded. There is no danger, however: if the isotopes had not been picked up by the wire, you would probably have breathed them in instead.

This method will let you determine which rooms in your house have more radon in them. Normally levels will be higher in a cellar or basement as the source of the radon is the earth below.

A suitable high-voltage generator was described as the 'Air Ionizer' mini project in the June 2009 edition of *Elektor* (http://www.elektor.com/071072). It is necessary to extend the design by a further two stages (two capacitors and two diodes) to obtain an output voltage of 5 kV.

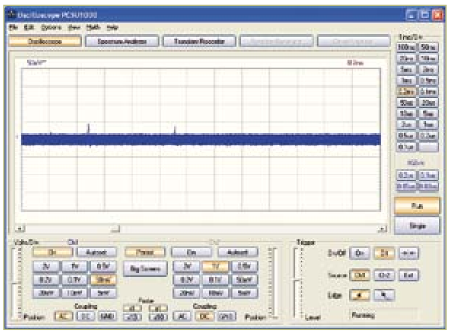


Figure 8. Readings over 30 minutes from a sample of galena.

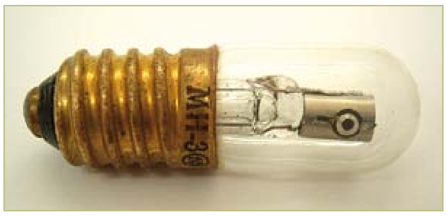


Figure 9. A gas discharge tube with a radioactive ioniser to aid starting.

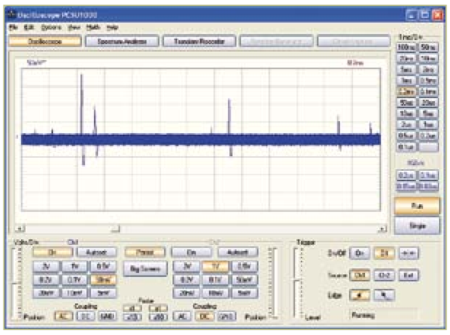


Figure 10. Readings taken from the gas discharge tube shown in Figure 9.

controls were in place can turn out to be radioactive sources. A well-known example is that certain gas discharge lamps and voltage regulator tubes rated under 100 V contain radioactive substances. The author had already had suspicions about an old Russian gas discharge lamp rated at 75 V/3 mA (**Figure 9**). There is a small metal cap welded on to the outer cover, beneath which a strange pill is visible. Beneath this is a tiny hole.

Taking readings over half an hour (**Figure 10**) revealed impulses with a particularly high energy: all the more surprising given that the radiation had had to penetrate the glass envelope of the tube.

Outlook

We have described the sensor and a simple amplifier for it. If the circuit is built into an enclosure, along with the comparator circuit and loudspeaker described in the text box, the result is a device that can be used in the field, for example to test minerals in a quarry. Combine the comparator with a digital counter, and the overall level of radioactivity can be measured. A sample-and-hold circuit could be added to record energy levels, and the results could be displayed as a kind of energy spectrogram.

Another possibility is to look at taking measurements from other samples over a long period. For example, potassium chloride is a very weak beta emitter. It would be interesting to see if the photodiode can detect it.

(110372)

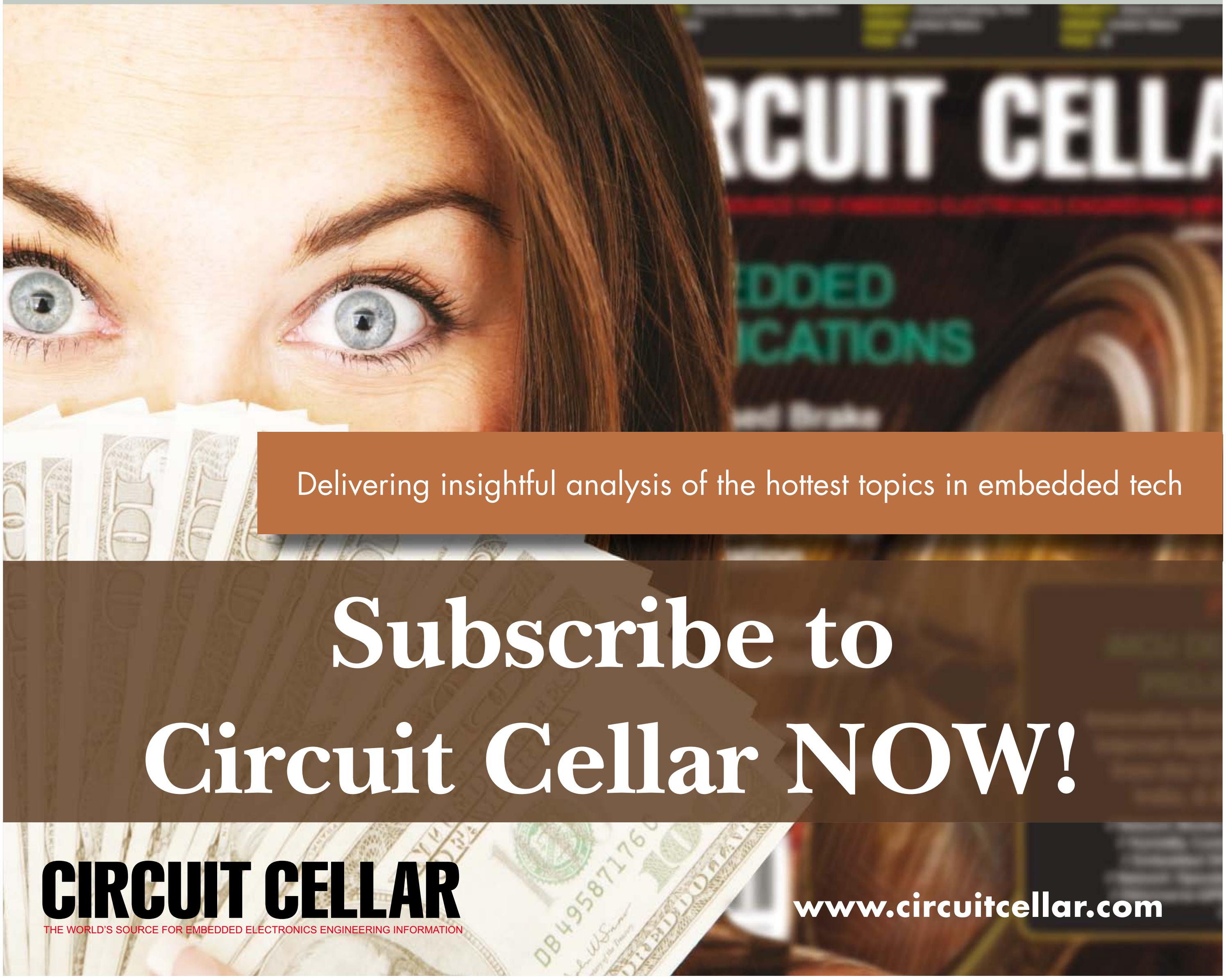
References and Internet Links

- **Maxim Application Note 2236:**
  Gamma-Photon Radiation Detector
  http://pdfserv.maxim-ic.com/en/an/AN2236.pdf
- **Erhan Emirhan and Cenap S. Özben:**
  PIN photodiode-based X- and gamma-ray detectors
  http://thm.ankara.edu.tr/tac/YAZOKULU/yazokulu6/dersler/06-09-2010/erhan-emirhan-cenap-ozben-pin-photodiode.pdf
- **C. W. Thiel:**
  An Introduction to Semiconductor Radiation Detectors
  www.physics.montana.edu/students/thiel/docs/Detector.pdf

Save

66%

off the cover price!



Delivering insightful analysis of the hottest topics in embedded tech

Subscribe to Circuit Cellar NOW!

CIRCUIT CELLAR

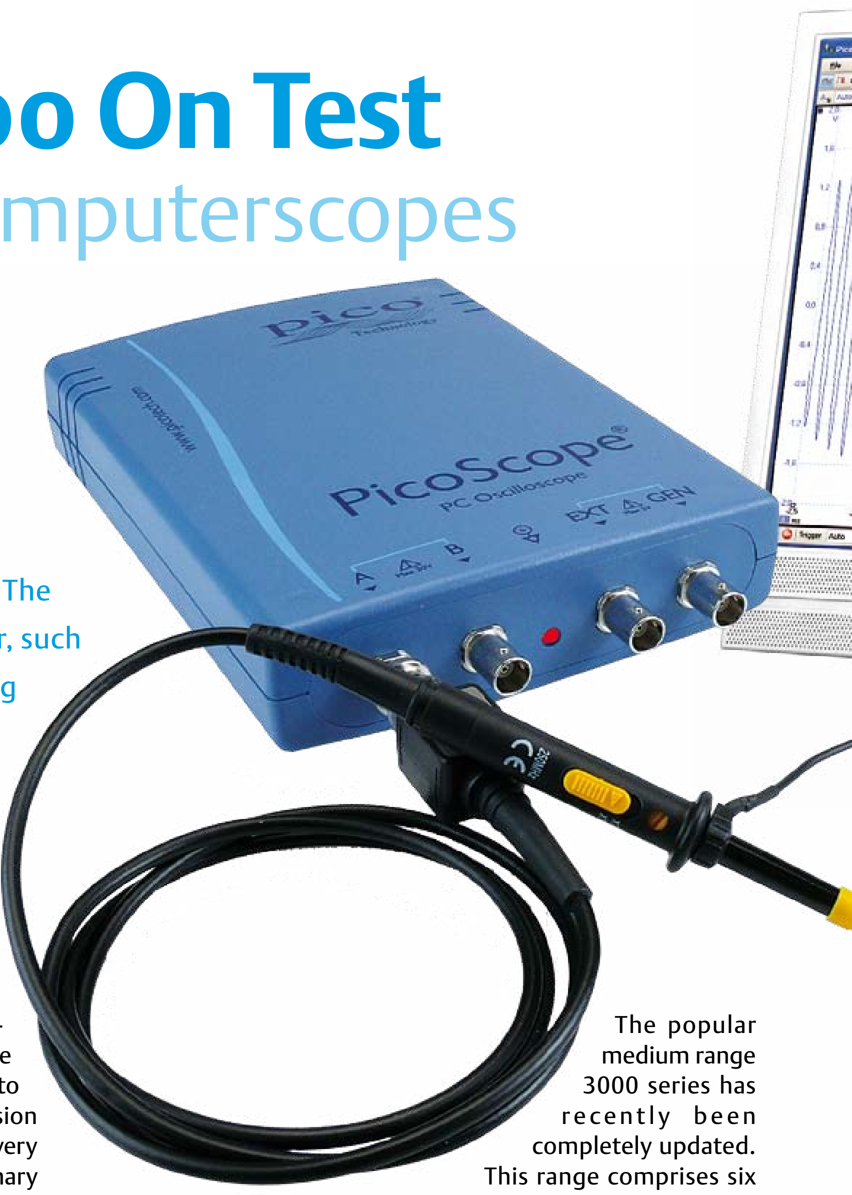
THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

www.circuitcellar.com

Picoscope 3000 On Test
A new series of computerscopes in practice

By Harry Baggen (Elektor Netherlands Editorial)

Pico Technology have completely updated their middle-of-the-range series of USB-oscilloscopes. The new PicoScope 3000 series has quite a bit to offer, such as, for a USB-powered scope, a very high sampling frequency of 500 Msamples/s and a built-in AWG/function generator. What is it like to use such a scope in practice? We spent some time using the type 3206B.



The popular medium range 3000 series has recently been completely updated.

USB oscilloscopes are forever becoming more powerful and more versatile. An electronics engineer is therefore increasingly tempted to buy a USB oscilloscope instead of a stand-alone version. The main motivation is not necessarily the potential to save some money, but more the advantages that a USB version could offer. The device has small dimensions and you have a very nice (computer) screen that is incomparable to that of an ordinary oscilloscope. In addition, a USB-scope combined with a laptop can be used for quite a while without being connected to an AC outlet and this combination has the further benefit of being completely isolated from the AC grid. And the supplied software often has a many more features which you are not likely to find on a comparable stand-alone version. But a stand-alone scope also has its advantages. The advantages of such an instrument are evident mainly at higher sampling frequencies, because the entire hardware of the instrument can be tuned to obtain the necessary operating speed. With a USB version you are stuck with the limitations of the data transfer between ‘scope and PC. This could be mitigated, for example, by adding more very fast memory to the scope hardware. This solution however, has the drawback that the hardware becomes increasingly more elaborate, with the consequence that the price advantage becomes smaller. But as already noted, a USB scope still has a few trumps in hand!

The new 3000 series from Pico

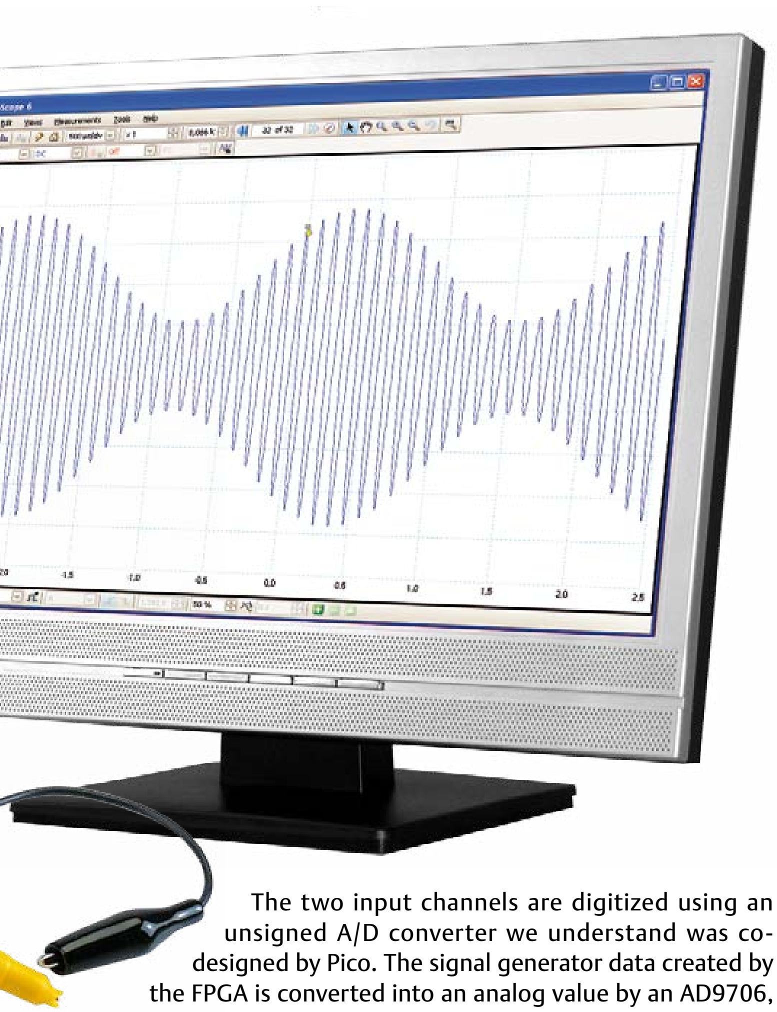
Pico Technology is a UK manufacturer who have specialized in the design and production of USB measuring instruments. They now have more than 20 years experience in that! Their range of USB oscilloscopes is very broad, starting with entry level models for about 125 pounds up to very fast models costing more than 13,000 pounds.

This range comprises six 2-channel models with prices ranging from about 500 to 1100 pounds. The entire series has a maximum sampling rate of 500 Msamples/s, and according to Pico — these are the most powerful USB-powered USB scopes available at the moment. The input bandwidth increases in steps and price from 60 to 100 and 200 MHz. In addition there are A- and B-versions, which have a different signal generator built in.

The A versions have a function generator which can generate a number of fixed waveforms, while the B version has an AWG (Arbitrary Waveform Generator) which allows the user to make his own waveforms. They all have a built-in buffer memory, which, depending on the model, ranges from 4 to 128 Msamples. Pico sent us the most expensive model from this series, the 3206B, which we used for a few days. This is our report.

Hardware

For starters, we could of course not resist the temptation to open the enclosure of the Picoscope, so that we could inspect the hardware it contains. **Figure 1** shows the opened enclosure, where the nicely screened input sections stand out. The heart of the scope consists of a powerful Spartan-6 FPGA from Xilinx, which takes care of the digital processing of the measuring signals as well as producing the signals for the signal generator. To this is connected a fast DD3 memory chip for storing the data samples.



The two input channels are digitized using an unsigned A/D converter we understand was co-designed by Pico. The signal generator data created by the FPGA is converted into an analog value by an AD9706, a 12-bit DAC from Analog Devices with 175 Msamples/s. Finally, a USB transceiver made by Cypress (CY7C68013A) takes care of the communications with the PC.

Software

Pico supply all their USB scopes with the same software, which has no limitations when it comes to features. The only limitations are those of the options and specifications of the connected hardware. The standard application is the oscilloscope screen, which shows the signals in a certain color on a white background. There is a special 'persistence' mode, which imitates the afterglow of a classic CRT oscilloscope screen (on a black background). The trigger options are very extensive. With the aid of a few buttons you can walk through the buffer memory in blocks and call up interesting signals.
There is also an 'XY' mode. In addition there is an FFT analyzer, which shows a continuous frequency analysis of the input signal. This allows from a choice of different types of sampling windows (Hamming, Blackman, etc.).
Furthermore, the software has the option of displaying and analyzing different types of serial data such as I²C, RS232/UART, SPI and CAN bus. This shows both the original measured signal as well as the decoded data on the screen at the same time. Using the generator button you can summon the menu for the built-in signal generator, where you can select the desired waveform and output voltage. With the B models from the 3000 series you also have the option of creating waveforms yourself using a graphical window (just draw the wave shape using the mouse) or from a CSV file.

In addition there are many more features and options, too many to discuss in detail, such as the option of making special scope profiles, the ability to make masks for, among other things, tolerance measurements in production processes, the various types of measurements that can be made on the displayed signals and carrying out mathematical functions.

In practice

The PicoScope 3206B is supplied with two probes in a storage pouch, a USB cable, an installation CD and a Quick Start Guide. The inconspicuous plastic enclosure has four BNC sockets on the front and one USB connector on the back. The installation of the software from the CD is done in five minutes, this went without any problems. After this the scope can be connected to the PC with the supplied cable and we can get started.
The first impression after starting the program is that of a plain appearance, all operating elements are accommodated in a few bars above and below the scope screen. The large screen format is immediately obvious, compared to our standard scopes in the lab! You really should start by reading the Help menu which explains the operation and features in detail, but as an electronics engineer you

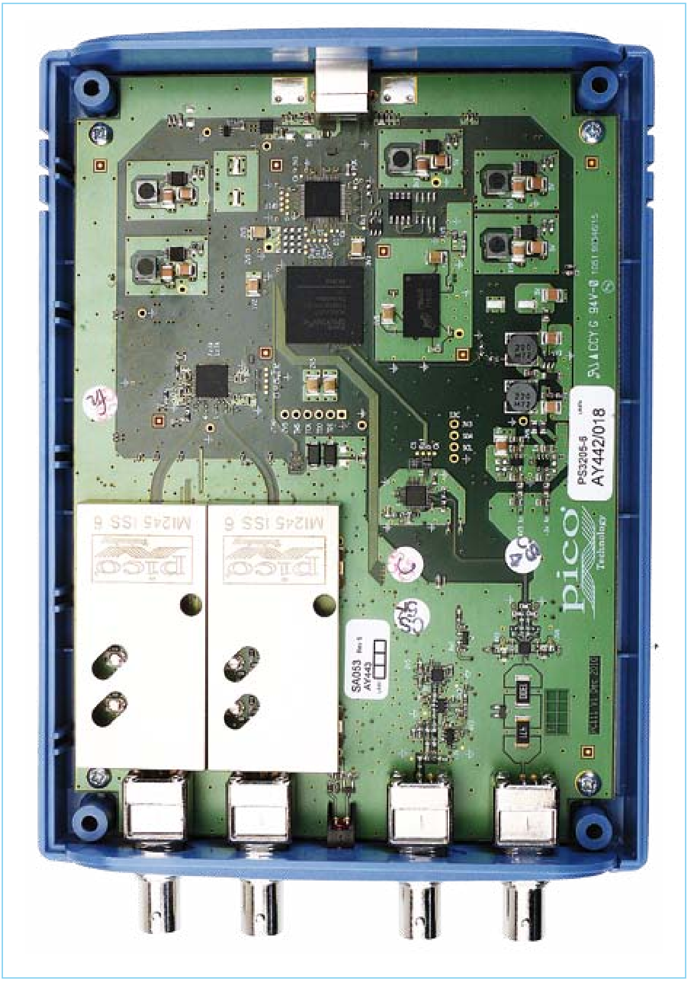


Figure 1. The inner workings of the new 3000 series from Pico. Note the shielded input sections.

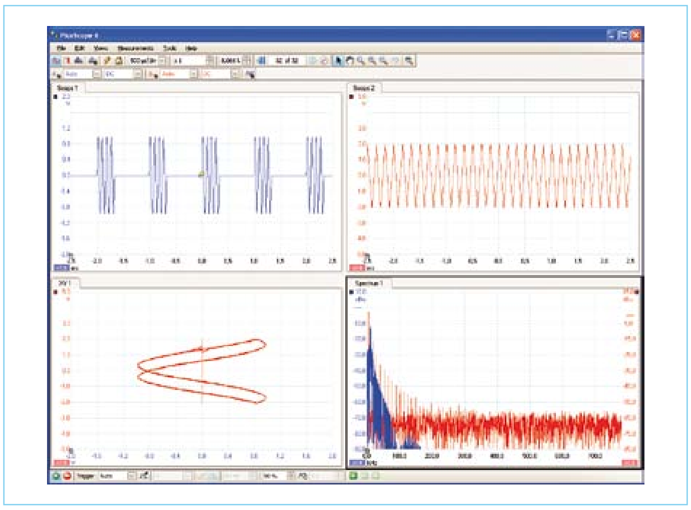


Figure 2. Here different windows of the main program have been opened.

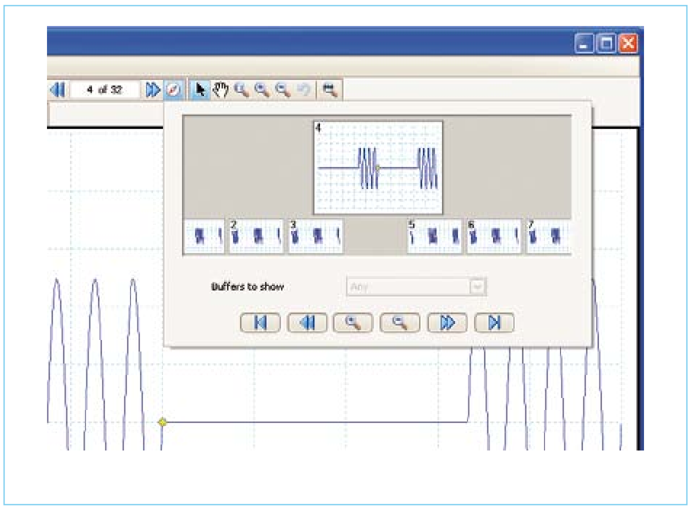
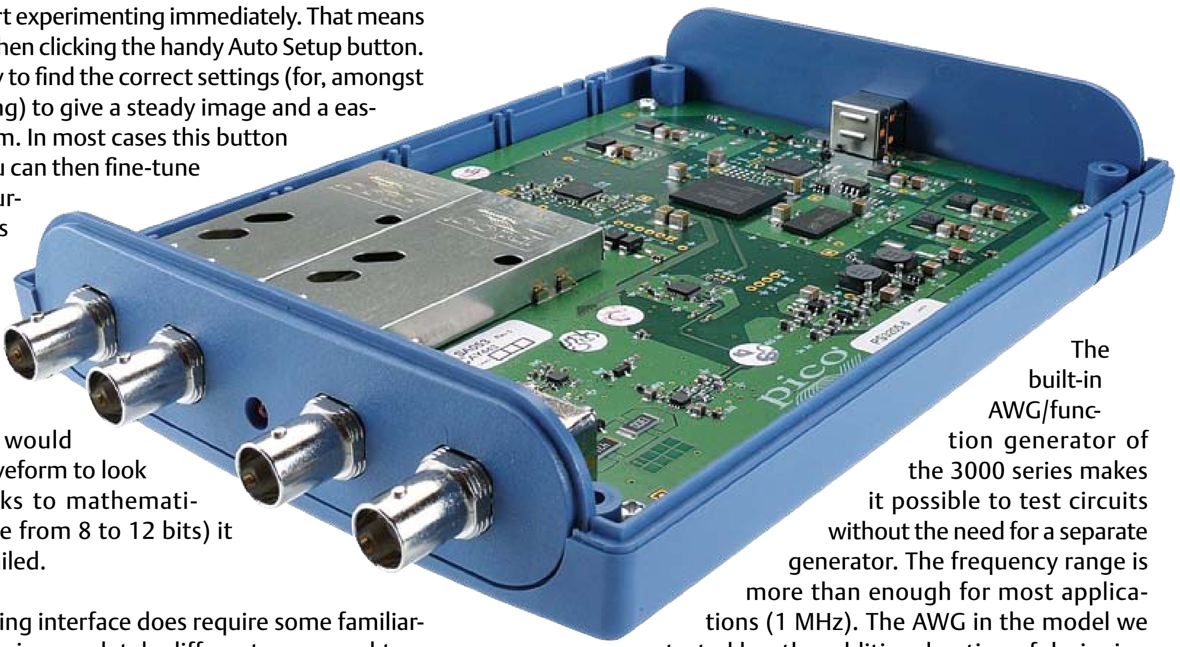


Figure 3. The contents of the buffer memory is easily observed thanks to the division in blocks which are shown as postage stamps in the overview.

would much prefer to start experimenting immediately. That means connecting a probe and then clicking the handy Auto Setup button. The software will then try to find the correct settings (for, amongst others, X, Y and triggering) to give a steady image and a easily recognizable waveform. In most cases this button appears to work well, you can then fine-tune some of the settings yourself. The screen reacts pleasantly quick to signal changes and this gives you the feeling that you're measuring in 'realtime'. With a resolution of eight bits and a large screen you would expect the displayed waveform to look quite course, but thanks to mathematical averaging (adjustable from 8 to 12 bits) it appears much more detailed.



The design of the operating interface does require some familiarity, because its structure is completely different compared to a normal scope. But after working a few days with this software it all becomes much more fluent. Pico have not tried to emulate the operating interface of a conventional scope and this is perhaps a good thing, otherwise it would have become a mixture of a few scope buttons next to a few menus for the special functions. Now you are obliged to think differently. The menus have been kept minimalistic, where only the most important functions are accessible with a single mouse click. All other things are hidden deeper into the menu structure, but most of it is fairly obvious. You can easily switch between one and two input channels or adding a new window with another signal or waveform analysis. The advanced trigger functions are very extensive and clearly documented. All the trigger options appear in a separate menu with a brief explanation and a small picture. It is easy to step through the buffer memory, which is displayed as a few screens in postage stamp format in a separate window, after which the desired data can be displayed in a large format.

The built-in AWG/function generator of the 3000 series makes it possible to test circuits without the need for a separate generator. The frequency range is more than enough for most applications (1 MHz). The AWG in the model we tested has the additional option of designing different waveforms graphically in a kind of mini editor. In this way it only took me five minutes to make a sine-burst signal, which is very suitable to judge the delay response and ringing of audio filters and loudspeakers. Although this is a very complex signal for getting a scope to trigger on consistently (without having the sine waves 'jumping around' all the time) I was quickly able to find an advanced trigger setting that worked perfectly.

Thanks to the high sample frequency of the new 3000 series (500 Msamples/s for one channel, 250 for two channels) it is possible, even at higher signal frequencies, to see the exact shape of the signal edges and look for any tendencies of oscillation or ringing. Especially in combination with the large buffer memory this gives many options for signal analysis. The maximum sample-rate for continuous (repetitive) signals (this is 10 Gsamples/s for the 3206A/B) is something we haven't tried, but it can be useful in some situations.

Excellent combination

The new 3000 series from Pico offers an excellent combination of a USB scope with a reasonably large sample frequency and a function generator/AWG with a useful frequency range. Starting from 399 pounds (excl. VAT) you have a 3204A with built-in function generator and 60 MHz input bandwidth. That gives you a fully-fledged scope with many features and a large screen (the computer monitor), which does not need its own power supply and is therefore very suitable in combination with a laptop.

Depending on your requirements for bandwidth, AWG and buffer memory size you can choose a more expensive version. The 3206B that we tested is very satisfactory in daily use. It is so good that is was difficult to return to using a conventional oscilloscope. With this 3000 series, Pico have made a versatile series of measuring instruments with state-of-the-art components, excellent performance and a great price/performance ratio.

(110268-I)

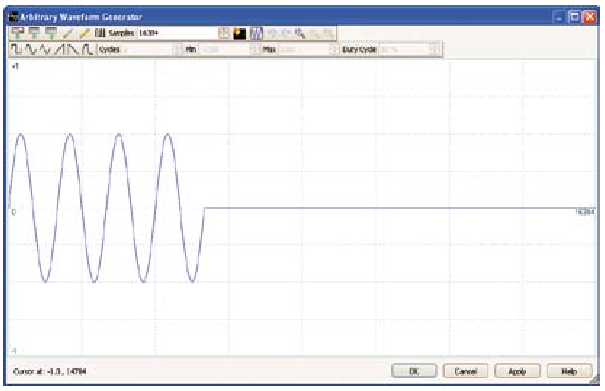


Figure 4. In the window for the Arbitrary Waveform Generator an unusual signal can be generated quickly.

More information:

www.picotech.Com/computer-oscilloscope.html

Advertisement

AP CIRCUITS
PCB Fabrication Since 1984

As low as...
$9.95
each!

Two Boards
Two Layers
Two Masks
One Legend

Unmasked boards ship next day!

www.apcircuits.com

VISA MasterCard PayPal IPC MEMBER ASSOCIATION CONNECTING ELECTRONICS INDUSTRIES BBB

USB Add USB to your next project. It's easier than you might think!

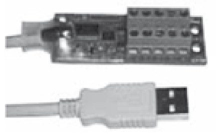
DLP-USB1232H: USB 2.0 UART/FIFO

HIGH-SPEED 480Mb/s

Only $28.95!

- Multipurpose: 7 interfaces
- Royalty-free, robust USB drivers
- No in-depth knowledge of USB required
- Standard 18-pin DIP interface; 0.6x1.26-inch footprint

DLP-IO8-G
8-Channel Data Acquisition



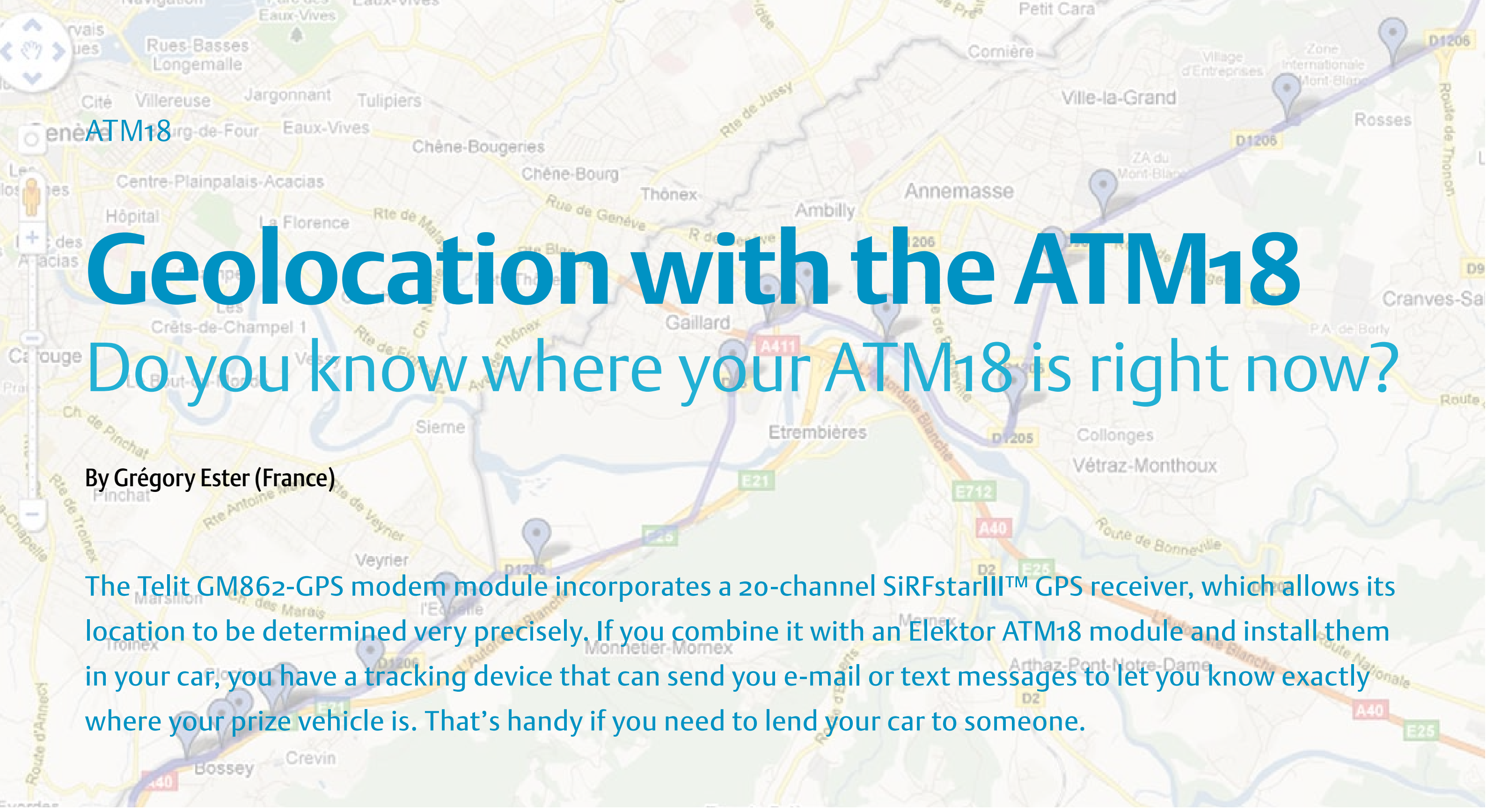
Only $29.95!

- 8 I/Os: Digital I/O
  Analog In
  Temperature
- USB Port Powered
- Single-Byte Commands

DLP-IOR4
4-Channel Relay Cable
DLP-TH1b
Temp/Humidity Cable
DLP-RFID1
HF RFID Reader/Writer
DLP-FPGA
USB-to-Xilinx FPGA Module

DLP Design

www.dlpdesign.com



Geolocation with the ATM18

Do you know where your ATM18 is right now?

By Grégory Ester (France)

The Telit GM862-GPS modem module incorporates a 20-channel SiRFstarIII™ GPS receiver, which allows its location to be determined very precisely. If you combine it with an Elektor ATM18 module and install them in your car, you have a tracking device that can send you e-mail or text messages to let you know exactly where your prize vehicle is. That's handy if you need to lend your car to someone.

The article 'Mobile, Text, Caller ID' published last month [1] described a project that uses the same GM862-GPS module, but we did not use the GPS functions of this versatile module in that project. This article remedies that situation. Incidentally, the project described in this article is self-contained, so you don't need to consult the previous article. However, we do use the same hardware here: an ATM18 module [2] with a two-wire LCD [3], which communicates with an OEM version of the GM862-GPS module [4]. The latter module is fitted on a prototyping board [5] along with antennas for GPS and GSM. See **Figure 1** for the interconnections between the modules. This hardware forms the platform for three related but different applications, which are summarized in **Table 1**. Each application has its own software, which you can download free of charge from the Elektor web page for this project [9].

Updating the GM862-GPS module

If you bought your GM862-GPS module some time ago, its firmware may be older than the current version (07.03.402), which means it needs to be updated. To find out which firmware version your GM862 module has, you need to connect it to a PC. To do this, connect RXI and TXO to the TxD (orange) and RxD (yellow) leads of a USB to serial converter (such as the FTDI converter [6]) instead of pins PC2 and PC3 of the ATM18 module. In your terminal emulator program (e.g. Hercules [7]), enter the command `AT+CGMR$0D`. If your modem replies with `Version 07.03.402`, you can skip the next three steps.

Use the following procedure to update the firmware:

1. Issue the command `AT+IPR=115200$0D` to configure the modem to 115,000 baud, and then restart the terminal emulator program with the same setting.
2. Next you have to register on the Telit website [8] in order to use their download page. What you need from the download page is an update tool called Xfp 2.13 and (of course) the update firmware. You can also order both files by sending an e-mail to the Telit support employee: Khaled.Chtourou@telit.com.
3. Connect power to the GM862-GPS prototyping board. If necessary, press and hold the On/Off button until the *STAT* LED goes out, to ensure that the module is switched off. Launch Xfp 2.13, configure the serial port number, and set the speed to 115,200 baud. Then click *Browse* and navigate to the update file on your hard disk. Next click *Program*, which causes the message *Linking...* to appear at the bottom of the window. The progress bar will also start blinking (see **Figure 2**). Now switch on the modem. This works the same way as with your mobile phone: just press the button briefly. The green *STAT* LED will light up, and the update will start. You should see the progress bar advancing to the right (**Figure 3**). When the update is finished, the *STAT* LED goes out and the message shown in **Figure 4** appears. Click *OK*, press the On/Off button on the modem prototyping board once again, and issue the command `AT+CGMR$0D` again. You should receive `07.03.402` in reply, which indicates that your GM862-GPS is now operating with this version of the firmware. Then reset the data rate of the modem to 9600 baud by issuing the command `AT+IPR=9600$0D`.

The GPS functions of the GM862-GPS module

The GM862-GPS allows you to collect all sorts of data related to its geographic location. You can request this data by sending AT commands, but there is also another option. The prototyping board has an output labeled EMMI_TX, and the same geographic data is sent out over this port, independent of the AT command interface. The EMMI_TX port outputs data in frames with the NMEA0183 format, which is a standard for serial data

Elektor Products & Services

- ATM18 microcontroller board: # 071035-91
- ATM18 prototyping board: # 071035-92)
- 2-wire LCD module: # 071035-93
- Project software: # 110267-11.zip

Items accessible via www.elektor.com/110267

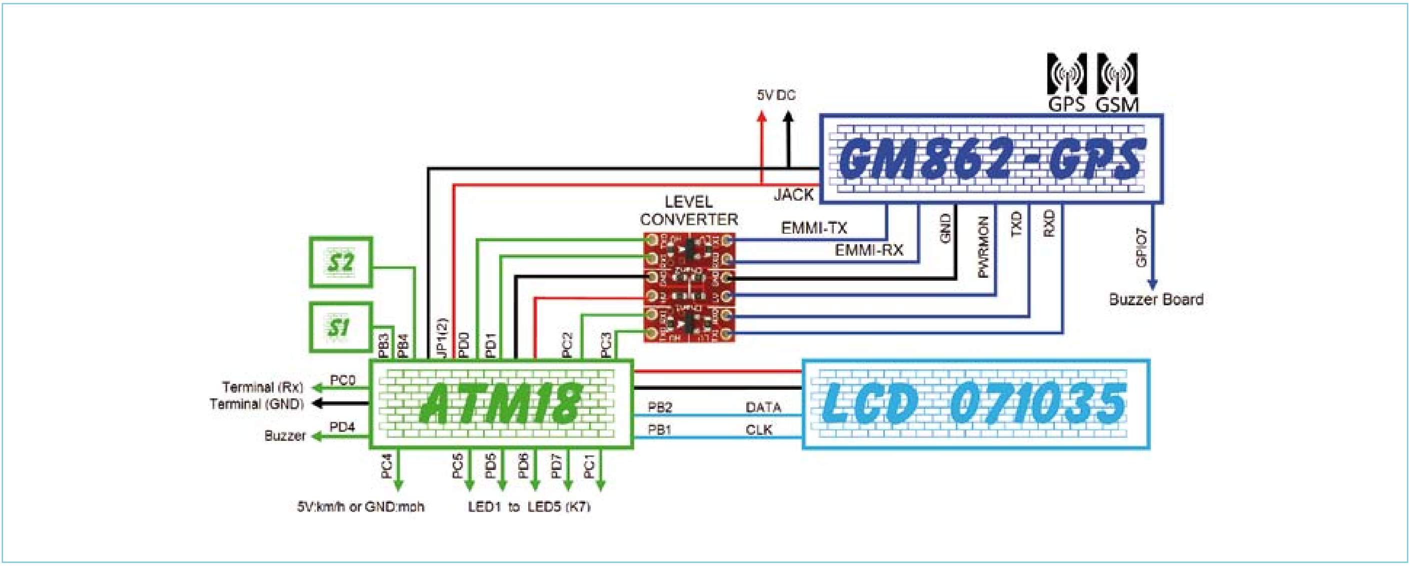


Figure 1. Block diagram and wiring of the various modules.

Table 1. Firmware files available for this project.

File	Function
110267-I_GM862-GPS_ATM18_P3.bas	Displays geographic data extracted from NMEA frames on the LCD
110267-I_GM862-GPS_ATM18_P4.bas	Sends geographic location data via text messages or e-mail
110267-I_GM862-GPS_ATM18_P5.bas	Collect BTS identification data (LAC and CI) and continually records signal quality and geographic position

communication in the maritime world.
To obtain data from this port, first press and hold the On/Off button to enable the GM862-GPS module. The *STAT* LED will start blinking at a 1-second rate, indicating that that the modem is not yet logged in to the network, but this does not stop it from outputting NMEA frames. By default the data stream emerges from the EMMI_TX port at 4800 baud and consists GGA, GSA, GSV and RMC frames in succession. What this all means is explained below.
The first of our three applications use the software in the file *110267-I_GM862-GPS_ATM18_P3.bas*, which parses GGA and RMC frame and outputs their data. Among other things, GGA frames contain the UTC time code, the latitude and longitude, the elevation above sea level, the number of satellites that are being received, and a checksum. We also want to know the speed and the date, which come from the RMC frames. This data is not valid unless signals are being received from at least four satellites. If this condition is satisfied, the program extracts the desired data from the frames, compiles it, and sends it to the LCD module. The data is displayed as shown in **Figure 5**, and the display is refreshed approximately once per second. If you press button 2, the display changes to the screen shown in **Figure 6**. Here the speed is shown in the third line. The speed is given in miles per hour if pin PC4 is tied to ground, or in kilometers per hour (see **Figure 7**) if it is connected to +5 V. The time zone is preset in the software to Central European time (CET), which is GMT (Greenwich Mean Time) + 1.

ATM18 telltale

The task of the second application is to use the GM862-GPS module to cause e-mail messages to be sent via port 25 of an SMTP server. Mobile telephones with GPRS capability can be used to send e-mail messages.

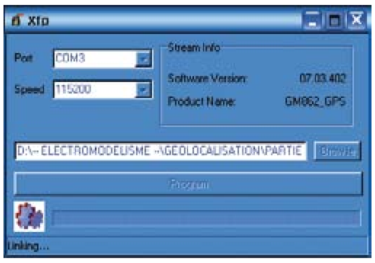

Xfp
Port COM3
Speed 115200
Stream Info
Software Version: 07.03.402
Product Name: GM862_GPS
D:\--ELECTROMODELISME --\GEOLOCALISATION\PARTIE
Linking...

Figure 2. Xfp *Ready* dialog.

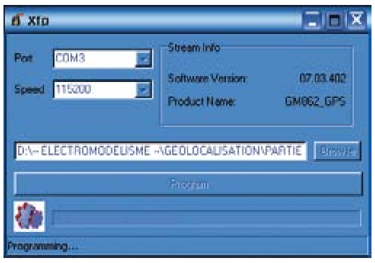
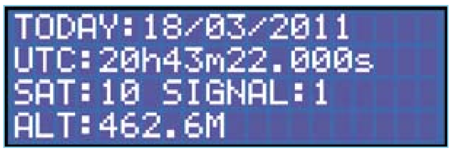

Xfp
Port COM3
Speed 115200
Stream Info
Software Version: 07.03.402
Product Name: GM862_GPS
D:\--ELECTROMODELISME --\GEOLOCALISATION\PARTIE
Programming...

Figure 3. Xfp *Go* dialog.

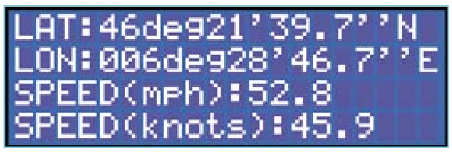
Xfp
Program Complete

Figure 4. Xfp *Done* dialog.



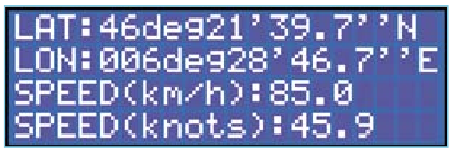
```
TODAY:18/03/2011
UTC:20h43m22.000s
SAT:10 SIGNAL:1
ALT:462.6M
```

Figure 5. The current time (GMT + 1) is 20:43.



```
LAT:46deg21'39.7''N
LON:006deg28'46.7''E
SPEED(mph):52.8
SPEED(knots):45.9
```

Figure 6. Where are you?



```
LAT:46deg21'39.7''N
LON:006deg28'46.7''E
SPEED(km/h):85.0
SPEED(knots):45.9
```

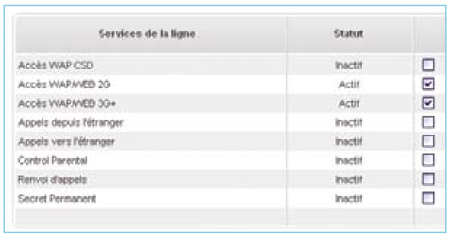
Figure 7. And where are you now?

However, some mobile phone subscriptions — especially the bargain variety — do not include GPRS service. If this is true in your case, you will have to ask your provider to activate this service.

Assuming your subscription includes GPRS service (see **Figure 8**), and you are able to sent e-mail, you need to install a different program to provide this capability: *110267-I_GM862-GPS_ATM18_P4.bas*.

Table 2 lists all of the data and configuration settings you need in order to send e-mail message with the GM862-GMS module. Here you are the sender who wishes to send messages to the receiver. After entering your PIN code, you can create a GPRS context using the commands listed below. In this listing, '`OK`' is always the response that you receive from the module.

```
AT+CPIN=7453<CR>
OK
AT+CGDCONT=1,"IP","internet68"
<CR>
OK
AT#esmtp="smtp.mail.yahoo.
fr"<CR>
OK
AT#euser="gpstracker74"<CR>
OK
AT#epassw="258369"<CR>
OK
AT#eaddr="gpstracker74@yahoo.
fr"<CR>
OK
AT#esav<CR>
OK
```



Services de la ligne	Statut	
Accès WAP CSD	Inactif	☐
Accès WAP/WEB 2G	Actif	✓
Accès WAP/WEB 3G+	Actif	✓
Appels depuis l'étranger	Inactif	☐
Appels vers l'étranger	Inactif	☐
Control Parental	Inactif	☐
Renvoi d'appels	Inactif	☐
Secret Permanent	Inactif	☐

Figure 8. GPRS service activated.

The commands listed above create the GPRS context and secure it. After this you can issue the following command to activate the context:

```
AT#SGACT=1,1<CR>
#SGACT: 10.189.67.153
OK
```

As you can see, the GM862-GPS module returns an IP address in response to this command. This is what you need in order to send e-mail messages. Use the following command for this purpose:

```
AT#EMAILD="my_address@my_
domain.uk","Test",0<CR>
```

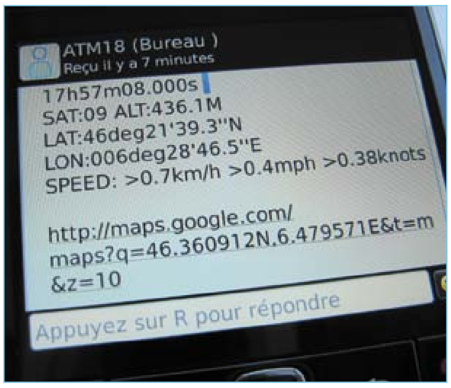
(Of course, you should replace `my_address@my_domain.uk` by your own e-mail address.) The modem responds with the prompt character **>**, after which you can send your message, terminated by `Ctrl-Z`.

```
> This is a test message.
[Ctrl-Z]
OK
```

Before you compile *110267-I_GM862-GPS_ATM18_P4.bas*, you must modify a number of constants as described in **Table 1**. Most of them are self-explanatory, but there are also four passwords, which must consist of

Table 2. Data necessary for sending e-mail messages via GPRS.

Parameter	Example data	Your data
SMTP server	smtp.mail.yahoo.fr	
E-mail address of the sender	gpstracker74@yahoo.fr	
E-mail address of the receiver	contact@adelek.fr	
E-mail subject	TEST	
E-mail content	Test GPRS feature, hello World !	
Access point name (APN) – depends on your service provider	internet68	
Login name for e-mail account (user ID)	gpstracker74	
Password for e-mail account	258369	



ATM18 (Bureau)
Reçu il y a 7 minutes
17h57m08.000s
SAT:09 ALT:436.1M
LAT:46deg21'39.3''N
LON:006deg28'46.5''E
SPEED: >0.7km/h >0.4mph >0.38knots

http://maps.google.com/maps?q=46.360912N,6.479571E&t=m&z=10

Appuyez sur R pour répondre

Figure 9. Geographic location data via text messaging.

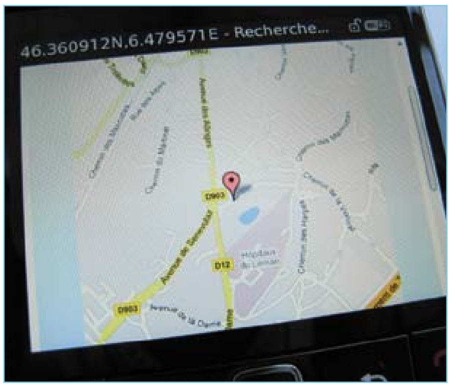


Figure 10. Where's my ATM18 board?

Listing 1

```
Const Decal = 1      'UTC + 1
'
Const Apn = «internet68»   'APN
Const Esmtp = «smtp.mail.yahoo.fr»      'EMAIL SENDER SMTP
Const Euser = «gpstracker74»            'EMAIL SENDER LOGIN
Const Epassw = «258369»                 'EMAIL SENDER PASSWORD
Const Eaddr = «gpstracker74@yahoo.fr»   'EMAIL SENDER NAME
'
Const Passw1t = «T090471»               'PASSWORD USER1
Const Passw1e = «E090471»
Const Phone1 = «0682834725»             'PHONE USER1
Const Email1 = «contact@adelek.fr»      'EMAIL USER1
'
Const Passw2t = «T180676»               'PASSWORD USER2
Const Passw2e = «E180676»
Const Phone2 = «06XXXXXXXX»             'PHONE USER2
Const Email2 = «stephanie.b@free.fr»    'EMAIL USER2
'
Const Code_pin = «7453»                 'SIM PIN
```

seven alphanumeric characters (excluding special characters). The purposes of these passwords are described below.

After you have compiled the program and loaded it into memory, you can use the unit as follows. First install it in the vehicle you want to be able to track, and then give the keys to a reliable volunteer with orders to go for a nice ride, while you stay at home to check things out.

There are two methods that you can use to keep track of where your accomplice is going with your wheels. The first method is to send a text message (SMS) with your previously configured password, which in our example is 'T090471'. A short while later you will receive a text message in response (see **Figure 9**) with the following information in sequence: the latitude and longitude, a web link for Google Maps, the date and time, the speed, and the number of satellites used to calculate this data. If you can access the Web from your mobile phone, click the Google Maps link to display a map with the position of your car marked by the familiar icon (**Figure 10**). In this URL `t=m` ('m' stands for map) means that a map will be displayed (but not photo imagery or a hybrid image) and `z=10` sets the zoom factor to a fixed value of 10.

The other method is to send a text message containing the other password ('E090471' in our example). In response you will receive the same information, but in this case it will come in an e-mail message sent to address `Email1` (see **Figure 11**), which in our example is `contact@adelek.fr`. The subject line of the e-mail shows the time when the information was calculated.

You can also authorise another user to do the same thing. You do this by giving them their own set of passwords ('T180676' and 'E189676' in our example). The other user will then receive responses on their phone (`Phone2`) or by e-mail (`Email2`).

Note that this program does not use the LCD module. Instead, the program status can be read from a set of five LEDs. Figure 1 shows how they should be connected.

Mapping cellphone base stations

Every cellphone is connected to the mobile telephone network by base transceiver stations (BTSs), and each BTS is surrounded by a cell within which the BTS can detect the mobile phones present in the cell. When you move about with your mobile phone, the connection is handed over from one cell to the next on so that you do not lose the connection, at least most of the time. In any case, every base transceiver station transmits its own unique identification code in the data sent to you mobile phone, and the approximate geographic position of the phone can be determined from this information. The identification code consists of two numbers: the Local Area Code (LAC) and the Cell Identity, which is the cell number. When you move closer to the BST of another cell, its takes over the connection and the LAC and CI data set changes. It's fairly easy to have the ATM18 module log this data set while it's moving through the network. Among other things, you can use this information to create a database of BTSs in your region.

The option of registering the identification codes is disabled by default. To enable it you must issue the command `AT+CREG=2`, which translates into 'enable network registration unsolicited result code with network cell identification data'. For this application you need a third program file, *110267-I_GM862-GPS_ATM18_P5.bas*, which contains the software that collects the LAC and CI codes of the mobile telephone network cells as you drive through them with the ATM and modem modules. Each time you change to a different cell, the program outputs data on pin PCO (see Figure 1) at 9600 baud, which can be sent to a terminal emulator program (such as

Table 3. Indicator LED functions with program 110267-I_GM862-GPS_ATM18_P4.bas.

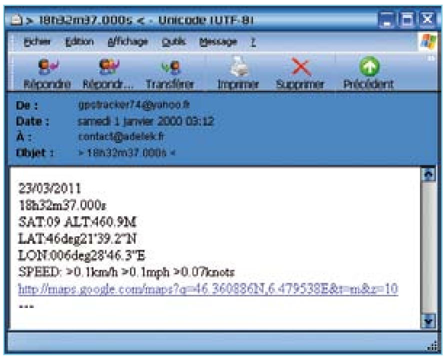
LED	Meaning
1	Fewer than four satellites are within range, the data in the GGA frames is not valid, or (if valid frames have been received) processing is currently in progress
2	Configuration commands are being sent to the GM862-GPS module, or the module does not see any network
3	Waiting for a text message
4	A text message is being sent in response to a geographic position request
5	An e-mail message is being sent in response to a geographic position request

Table 4. Indicator LED functions with program 110267-I_GM862-GPS_ATM18_P5.bas.

LED	Meaning
1	Fewer than four satellites are within range, the data in the GGA frames is not valid, or (if valid frames have been received) processing is currently in progress
2	Configuration commands are being sent to the GM862-GPS module, or the module does not see any network
3	Waiting for connection to a new base transceiver station (BTS)
4	Retrieving the identification data (LAC & CI) for a new BTS and the current signal quality data, and sending this data and the geographic position data to the terminal emulator program

Hercules) running in logging mode on your PC, so that the received data can be stored directly in a file. This data is structured as follows:

```
----------
Start with BTS: +CREG: 2,1,"296A","4437"
Signal Quality: +CSQ: 30.0
13/03/2011
16h16m15.000s
SAT: 05 ALT: 462.1M
LAT: 46deg21'40.9"N
LON: 006deg28'44.5"E
http://maps.google.com/maps?q=46.361347N,6.479033E&t=m&z=10
----------
----------
BTS: +CREG: 1,"296A","28D1"
Signal Quality: +CSQ: 21.0
13/03/2011
16h17m02.000s
SAT: 08 ALT:464.4M
LAT: 46deg21'40.8"N
LON: 006deg28'44.5"E
http://maps.google.com/maps?q=46.361320N,6.479023E&t=m&z=10
----------
----------
BTS: +CREG: 1,"296A","A2B3"
Signal Quality: +CSQ: 21.0
13/03/2011
16h21m52.000s
SAT: 08 ALT:448.7M
LAT: 46deg21'33.9"N
LON: 006deg27'48.8"E
http://maps.google.com/maps?q=46.359409N,6.463560E&t=m&z=10
----------
etc.
```


> 18h32m37.000s < - Unicode (UTF-8)
Fichier Edition Affichage Outils Message ?
Répondre Répondr... Transférer Imprimer Supprimer Précédent

De : gpstracker74@yahoo.fr
Date : samedi 1 janvier 2000 03:12
À : contact@adelek.fr
Objet : > 18h32m37.000s <

23/03/2011
18h32m37.000s
SAT:09 ALT:460.9M
LAT:46deg21'39.2"N
LON:006deg28'46.3"E
SPEED: >0.1km/h >0.1mph >0.07knots
http://maps.google.com/maps?q=46.360886N,6.479538E&t=m&z=10

Figure 11. Geographic location data via e-mail.

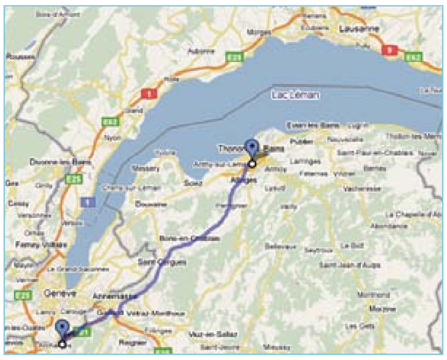


Figure 12. A 25-mile trip along 27 BTSs.

Here again, this program does not use the LCD module; instead you can monitor what's happening by observing the states of four LEDs as described in **Table 4**. We tested this program with a trip extending over a distance of about 25 miles (40 km)

Internet Links

[1] www.elektor.com/110139

[2] www.elektor.com/071035

[3] www.elektor.com/071148

[4] www.telit.com/en/products/gsm-gprs.php?p_ac=show&p=7

[5] www.sparkfun.com/products/281

[6] http://www.elektor.com/products/kits-modules/kits/080213-71-cc2-avr-board.35.441731.lynkx

[7] www.hw-group.com/products/hercules/index_en.html

[8] www.telit.com/en/products/download-zone.php

[9] www.elektor.com/110267

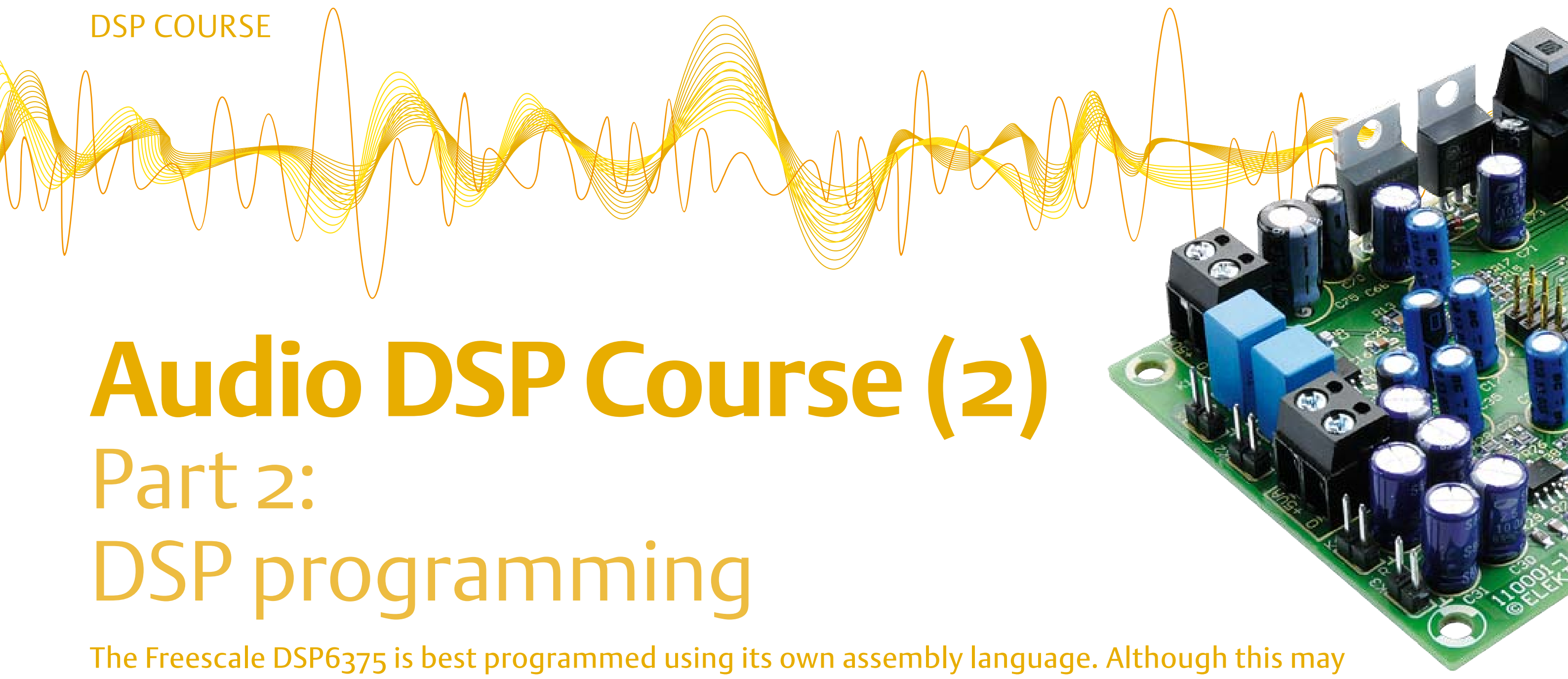
(see **Figure 12**). We identified 27 base station along this route (see **Table 5**).

The author hopes that you find this design interesting and useful, and he would appreciate receiving reports on your results (via e-mail at tellme@adelek.fr).

(110267-I)

Table 5. Base transceiver station identification codes (list of 10 of the 27 ID data sets collected over a distance of 25 miles along the route shown in Figure 12).

LAC	CI
296A	1D43
296A	B93B
296A	F11D
296A	94CC
296A	B93B
296A	94CC
296A	B93B
296A	922B
296A	94CC
296A	C2B8



Audio DSP Course (2)

Part 2: DSP programming

The Freescale DSP6375 is best programmed using its own assembly language. Although this may sound outdated — an undesirable leftover from early days of digital computers — this is not the case with the DSP56374. In part this is because the parallel data transactions typical of signal processing arithmetic cannot be handled more easily in a high-level language, and in part because the code generated using a high-level language is not as well optimized at the execution level as code generated using assembly language, especially since most DSP programs have very simple structures. Here the main hurdles in the programming process arise from stringent requirements for precision, and even high-level languages offer no help in this regard.

By Alexander Potchinkov (Germany)

The assembly language of the DPS563xx family is relatively easy to learn. It has high structural consistency and is based on a well conceived programming model, which also benefits from what manufacturers can learn from existing DSPs before launching their own products, and especially what they can learn from experience with their own processors.

You will need three PC-based programs for software development: an assembler, a simulator and a debugger. They are available as individual programs or as software tools in a package called an integrated development environment (IDE). All you essentially need is an assembler, but working without a simulator and debugger means doing without the most important tools for finding and fixing errors in DSP programs. The assembler converts source code into object code, while the debugger and simulator help you locate and correct errors in the program. The latter tools are very similar and have nearly the same functions and features, with the basic difference that a debugger needs to have a real DSP available, while a simulator does not.

Assembler

The starting point is a text file containing a DSP program written in assembly language. This text file may be generated using any desired editor. Each instruction may have up to six separate fields, separated by spaces:

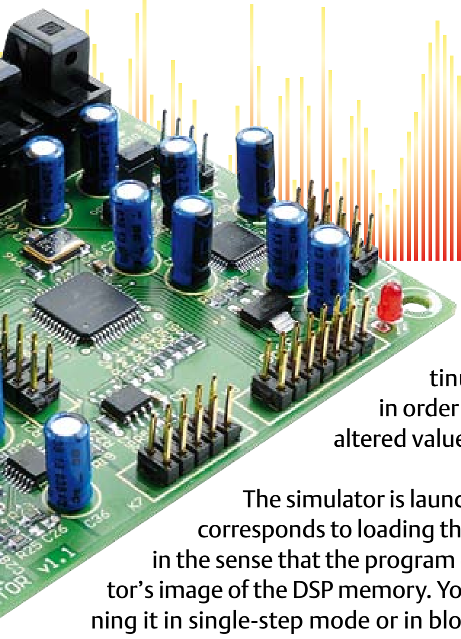
A comment begins with a semicolon. From the source code file, the assembler generates a file containing the object code. This file has the extension `.cld`. You can use this file with the simulator, or you can use the debugger to load it into the DSP and run it in the DSP.

Simulator

The simulator is a very useful tool for software development. In a manner of speaking, the simulator is an emulation of the DSP that runs in the PC environment at reduced execution speed and lacks the peripheral devices of the DSP. The emulation includes not only the data path of the DSP, but also all of the interfaces and the entire memory, as well as the interrupt system. As the simulator also emulates the pipelining, it allows the number of DSP processor clock cycles necessary for any desired software routine to be counted, which is of course important information for real-time processing.

Obtaining this information directly from the actual DSP program would be a very laborious task. The simulator can also be used for debugging, since the virtual registers and memory locations of the DSP can be read and written for each step of a program. For example, you may find that incorrect program behavior is caused by an incorrect value in a register, or you can change the value in a register at a particular point in the program and allow the program to con-

Label	Operator	Operands	X Bus Transfer	Y Bus Transfer	Comment
`Label`	`mac`	`x0,y0,a`	`x:(R0)+,x0`	`y:(r4)+,y1`	`; Mac-Operation`



tinue executing from that point, in order to see how it behaves with the altered value.

The simulator is launched with the object code. This corresponds to loading the program into the actual DSP, in the sense that the program code is loaded into the simulator's image of the DSP memory. You can test the program by running it in single-step mode or in block mode. For this purpose the simulator offers conditional and unconditional breakpoints, instruction and processor clock cycle counters, a single-line assembler, and the option of displaying all memory and register contents either individually or block wise and saving them as ASCII files.

One of the major uses of the simulator is running it in file I/O mode. In this mode you can test the signal processing characteristics of a DSP program by using test data in a file (the test signal) as input and writing the output data from the program (the processed signal) to another file, which can be analyzed using a standard program.

To illustrate how to use the simulator, here we present a simple example based on the source file `fileio.asm`, which the assembler converts into the object file `fileio.cld`.

```
   org x:$0 ;
simdatain   ds 1  ; memory location for input data
simdataout  ds 1  ; memory location for output data
   org p:$100
start move x:simdatain,x0  ; x0 <- input data
   move x0,x:simdataout    ; x0 -> output data
   jmp start               ; repeat process
```

In the DSP program the content of DSP memory location `simdatain` is copied to DSP memory location `simdataout`, with register x0 being used for temporary storage. This process is repeated indefinitely, which is achieved by including the jump instruction `jmp start`. An input file named `infile.dat` is generated for running the program in simulator mode.

The simulator commands listed below read data from the input file and generate an output file named `outfile.dat`, which in this case should contain the same data as the input file.

```
(1)   reset s  ; reset simulator
(2)   input off   ; reset all input files
(3)   output off  ; reset all output files
(4)   load fileio.cld   ; load the dsp program
(5)   input #1 simdatain infile.dat -rh
         ; input from infile.dat
(6)   output #1 simdataout outfile.dat -rh -o
         ; output to outfile.dat
(7)   break EOF   ; stop when an input file reaches
                    end-of-file
(8)   go        ; execute program
```

The functions of the individual simulator instructions are:

- The instructions in lines 1 to 3 reset the simulator and close any currently open input and output files.
- Line 4 loads the DSP program.
- Line 5 loads the input file. The option –rh indicates that the input data is to be read as hexadecimal numbers. The alternative option –rf can be used to read fractionals, which are important for signal processing.
- Line 6 opens the input file. The option –o specifies that an existing file with the name given in the instruction should be overwritten.
- Line 7 tells the simulator that the process should be stopped when the end of the input data is reached.
- Finally, line 8 initiates the execution of the DSP program in simulator mode.

The simulator instructions can be saved in a command file.

Debugger

The debugger does essentially the same thing as the simulator, but with the program running in the actual DSP instead of running on a simulated DSP. You can use the debugger to load programs into the DSP and run them in single-step mode or execute them with breakpoints. The debugger can put the DSP in a special mode in which all of the registers and memory contents can be read or written. The execution speed is much faster than with the simulator, since the program runs on real hardware. Instead of using the file I/O method with the test and response signal files, you can test your DSP programs via the audio interfaces, for example by using a wave editor to generate test signals.

Using the software

Figure 1 illustrates the relationships between the PC programs (shown in rectangular boxes with grey shading) and the files needed and/or generated in this context (shown in boxes with rounded corners). The starting point is the source code, which is converted into object code by the assembler. When you run the assembler, you can use the option `-l` to specify the creation of a list file, which often comes in handy. In our case we need absolute object code, which you can generate by specifying the assembler option `-a`. The object code can be fed into three different programs: the debugger (with a DSP connected to the PC), the simulator, or a program called *srec*, which we haven't mentioned yet. You can use this program to generate S records, which are necessary for programming a bootstrap PROM. Commonly used programming devices can handle these S records.

Software develop environments

You need two things for DSP software development: the development software tools and an adapter to connect the debugger pro-

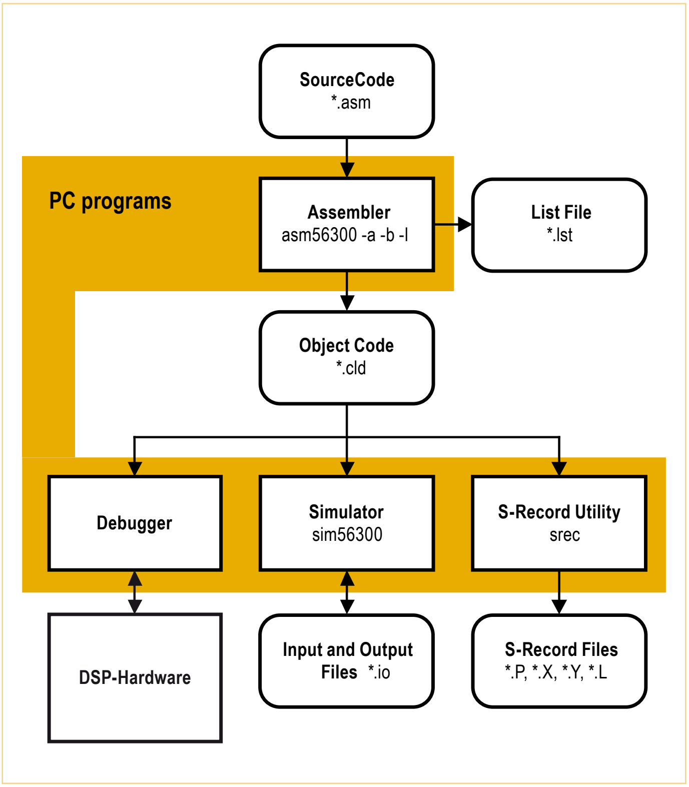


Figure 1. Program and file relationships in the Freescale software development environment.

gram running on the PC to the DSP.

• Freescale Suite 56

Software

The software, consisting of an assembler, a linker, a simulator and a debugger, can be downloaded free of charge from the Freescale website [1]. The simulator and the debugger are available in two forms: either with a command-line user interface or with a graphic user interface (GUI56300 for the simulator and GDS56300 for the debugger). The software has a relatively short learning curve. **Figure 2** shows the software components of Suite 56. Suite 56 is especially suitable for use in a team environment, thanks to the presence of a linker and various libraries.

Adapter

The link to the DSP is provided by the parallel port of the PC and an adapter, which can be obtained from Freescale [2] or (in identical form) from Macraigor under the name 'Wiggler' [3]. You can also build the adapter yourself, since all that is needed is some bus drivers and a few other components. Detailed designs and documentation for this can be found on the Web, such as the UniDSP56 from Gerrit Buhe [4]. If your PC lacks a parallel port, you can fit a plug-in PCI card with a parallel port.

• Freescale Symphony Studio

Software

Freescale also makes the assembler, linker, C compiler, simulator and debugger available as free software plugins for the Eclipse development environment [5]. However, configuring these plugins to create a software development environment is not well documented, and if you are not familiar with Eclipse it may cause you trouble. On the other hand, it's worthwhile to learn how to use modern software development environments, since they can be used with many different types of processor.

Adapter

As with Studio 56, the link to the DSP is provided by the parallel port of the PC and an adapter, which can be obtained from Freescale [2] or (in identical form) from Macraigor under the name 'Wiggler' [3]. Another option is to use a USB adapter, such as the usbWiggler from Macraigor [6] or USB-EMU from Domain Technologies [7].

• Domain Technologies BoxView

Software

Domain Technologies produces a software debugging tool that is sold under the name BoxView [8]. You can use this tool in combination with the free assembler and simulator tools from Freescale (asm56300 and sim56300).

Adapter

A USB adapter [7] is available from Domain Technologies.

• Domain Technologies EVM package debugger

A BoxView debugger tool (Bv30xEvm.exe) for use with Freescale's evaluation module (EVM) boards for DSP563xx digital signal processors is available from Domain Technologies as a free download [9], if you have not already obtained a copy in an EVM package. This tool is especially user-friendly and is the author's favorite debugger. **Figure 3** shows an example screen shot from a debugging session using the BoxView debugger tool. Along with a panel displaying the source code, you can see other panels that display the contents of registers and memory. Memory contents can also be shown in graphic form, which can often be very helpful.

Adapter

One option is to obtain an EVM board (sometimes available second-hand at a friendly price) and use it as an adapter for a separate DSP board.

Along with the usual commercially available adapters and DIY adapters that connect to the parallel port of a PC, the author and Elektor are currently collaborating on the development of two inexpensive USB adapters intended to help Elektor readers get up to speed in the world of digital audio signal processing, in combination with the hardware for this DSP course.

DSP code and audio loop

All of the DSP programs in this series of articles have the same high-level structure, consisting of a framework program and an audio loop.

The framework program is largely the same for all of the applications; it contains the setup code and the code for operating the DSP peripheral interfaces. In addition it performs the initialization of the SRC and the application, such as clearing state memories, to ensure that the program always starts under the same conditions. The audio loop contains the code for audio synchronization and digital signal processing.

Two sets of input and output channels are configured in the framework program. They are available in the form of a two-channel

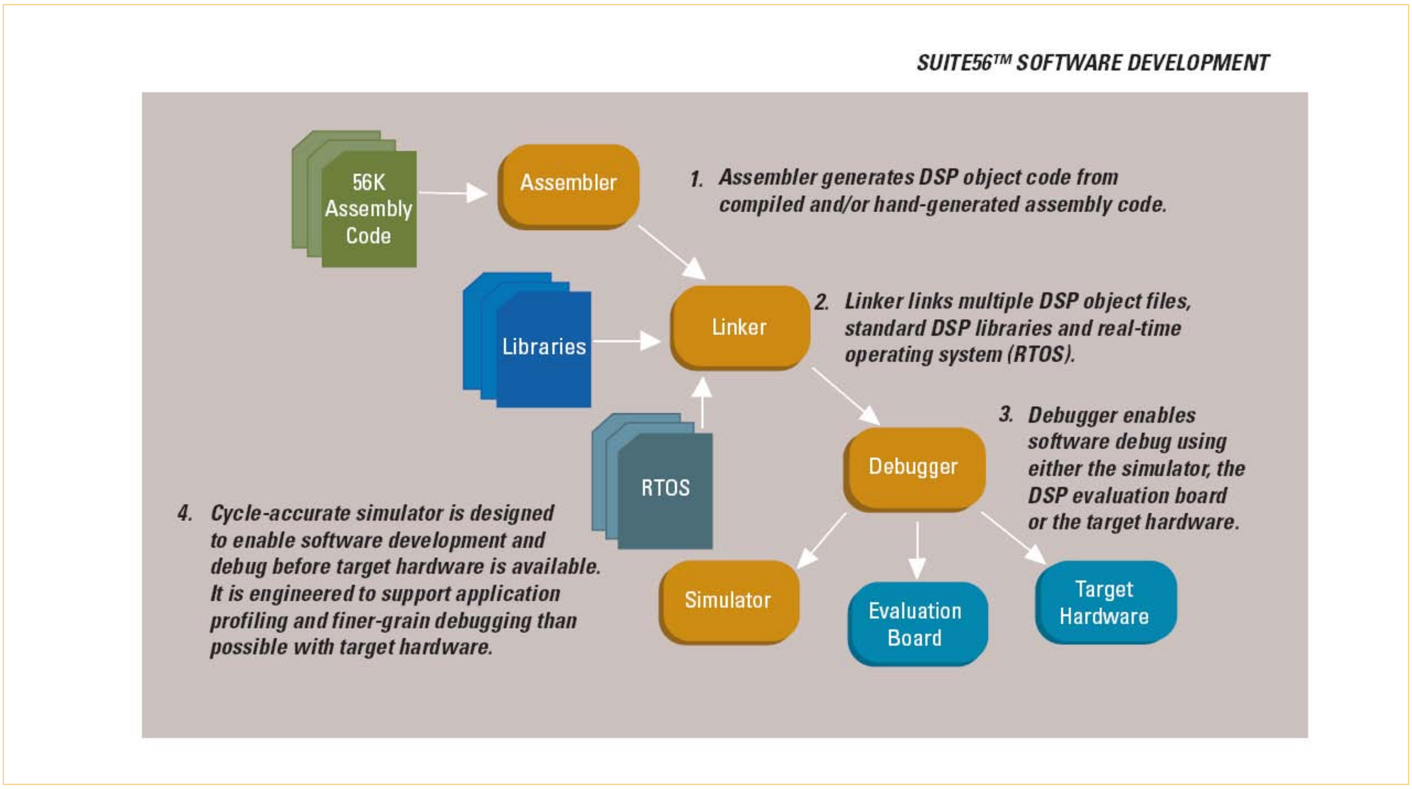

SUITE56™ SOFTWARE DEVELOPMENT
56K Assembly Code
Assembler
1. Assembler generates DSP object code from compiled and/or hand-generated assembly code.
Libraries
Linker
2. Linker links multiple DSP object files, standard DSP libraries and real-time operating system (RTOS).
RTOS
Debugger
3. Debugger enables software debug using either the simulator, the DSP evaluation board or the target hardware.
4. Cycle-accurate simulator is designed to enable software development and debug before target hardware is available. It is engineered to support application profiling and finer-grain debugging than possible with target hardware.
Simulator
Evaluation Board
Target Hardware

Figure 2. Suite 56 software tools (image courtesy of Freescale).

receiver on audio data port RX0 and a two-channel transmitter on audio data port TX0. The DSP program needs two memory buffers, which are used as audio data buffers. They are set up in X RAM and provided with pointers to the base addresses of the buffers:

```
TxBuffBase   EQU   $000000  ; BaseAddress of TX-Buffer (X)
RxBuffBase   EQU   $000010  ; BaseAddress of RX-Buffer (X)
TXPTR        EQU   $000020  ; Address of Tx-BufferPointer (X)
RXPTR        EQU   $000021  ; Address of Rx-BufferPointer (X)
```

We also need a flag called the left/right flag (`LRFlag`), which is polled in the main loop of the program in order to perform audio data synchronization.

```
LRFlag   EQU   $000022  ; Address of Left/Right-Flag
RightRx  EQU   0        ; Bit Position in Left/Right-Flag
```

With two-channel audio data transmission the left channel sample is transmitted in the first slot and the right channel sample is transmitted in the second slot, and the DSP counts these slots as 0 and 1. For this purpose the DSP audio interfaces are operated in network mode, which can be configured for two channels. In network mode the interrupt system of the audio interfaces generates an interrupt when the last slot has been read in. The interface can manage up to 32 audio channels for each audio data port, but here we need only two of these channels.

This means that the last slot holds data for the second channel, which is the right channel. The left/right flag, which has a value of 1 when data for the right channel has been received, is set in an interrupt service routine (ISR) triggered by the Last Slot interrupt.

In the initialization section the two audio buffers are filled with zeros and the synchronization flag is reset:

```
org    x:TxBuffBase
       dc 0
       dc 0
       endm
   org    x:RxBuffBase
       dc 0
       dc 0
endm
   org    x:LRFlag
       dc 0
```

The program uses six long-interrupt ISRs to operate the audio interfaces. The corresponding `jsr` instructions and ISR base addresses are entered in the interrupt vector table.
Now we're ready to execute the DSP code. The first task is to configure the audio interfaces. The settings for this must be selected individually because they depend on the hardware that is used. The most important setting determines whether the DSP acts as an audio master or as an audio slave. If the DSP is an audio master, the audio data clocks must be derived from the DSP processor clock. In this situation it is advisable to make the clock oscillator frequency an integer sub-multiple of the master audio clock frequency. For example, if the master audio clock frequency is 24.576 MHz you can use a standard 6.144 MHz crystal and configure the DSP clock PLL with a multiplication factor of 24, yielding a DSP processor clock frequency of 147.456 MHz. Now the audio interface clock signals can be derived by dividing the processor clock signal by 6 to obtain the master clock signal and by 3072 to obtain the sampling rate signal (48 kHz), which clocks the left/right channel data. If the DSP should

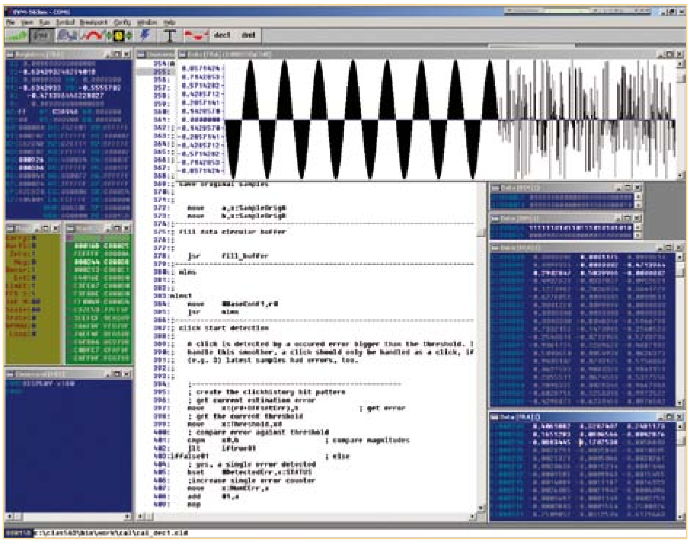


Figure 3. Domain Technologies BoxView debugger window.

act as an audio slave, the required clock signals can be generated by an A/D converter or a digital audio receiver acting as a master. In order to configure the audio interfaces, you also need to specify the network mode, the receiver and transmitter to be used, the data format, the interrupts to be used, and so on.
The audio loop starts by polling the synchronization flag:

```
AudioLoop
   jclr  #RightRx,X:LRFlag,*   ; right word received?
   bclr  #RightRx,X:LRFlag     ; reset synchronization flag
```

Here the `jclr` instruction performs the polling. The polling loop is executed repeatedly until the flag is non-zero. Iterative polling of the flag is achieved by using an asterisk (`*`) as the jump target. The subsequent `bclr` instruction resets the flag to zero so that it can be set again by the corresponding ISR.
The following code segment first writes the two words in the receive buffer to the two accumulator registers `a` and `b`.

```
   move  x:RxBuffBase,a     ; Left Channel ->a
   move  x:RxBuffBase+1,b   ; Right Channel ->b

; insert signal processing here

   move  a,x:TxBuffBase    ; a-> Left Channel
   move  b,x:TxBuffBase+1  ; b-> Right Channel
   jmp   AudioLoop
```

After this the audio data can be processed by the DSP code. At the end of this processing, the six signal values generated by the DSP code are written to the transmit buffer. In the example code, the two accumulator registers `a` and `b` are read for this purpose. In the absence of any signal processing, the DSP passes the received data and a sampling clock to the output with no delay. The final instruction jumps back to the polling loop for the synchronization flag.

In the next installment...

This completes our introduction to DSP programming. In the third installment of this series we will describe the hardware specifically developed for this course.

(110002-I)

Internet Links

[1] **Freescale Suite 56 software:**
www.freescale.com/webapp/sps/site/prod_summary.jsp?code=CW-SUITE56&fsrch=1&sr=4

[2] **Freescale adapter:**
www.freescale.com/webapp/sps/site/prod_summary.jsp?code=DSPCOMMPARALLEL

[3] **Macraigor Wiggler:**
www.macraigor.com/wiggler.htm

[4] **UniDSP56 DIY adapter design from Gerrit Buhe:**
www.unidsp56.de/downloads.html

[5] **Freescale Symphony Studio Software:**
www.freescale.com/webapp/sps/site/prod_summary.jsp?code=SYMPH_STUDIO&fsrch=1&sr=16

[6] **Macraigor usbWiggler:**
www.macraigor.com/usbWiggler.htm

[7] **Domain Technologies adapter:**
www.domaintec.com/usbemu.html

[8] **Domain Technologies BoxView debugger tool:**
www.domaintec.com/BoxView.html

[9] **Domain Technologies EVM package debugger tool:**
www.domaintec.com/FSsoftware.html

About the author

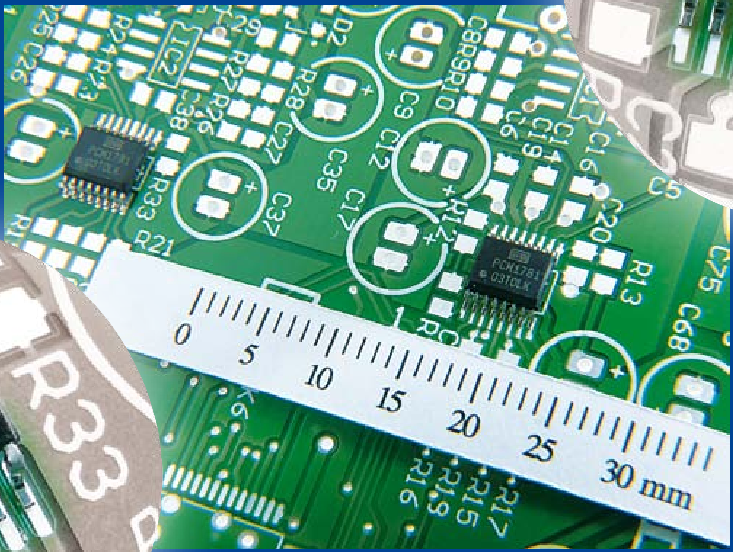
Alexander Potchinkov holds the Chair of Digital Signal Processing at the Technical University of Kaiserslautern (Germany) and runs an engineering consultancy specializing in audio signal processing. Aside from DSPs and DSP algorithms, his interests include tube amplifiers and SPICE simulations.

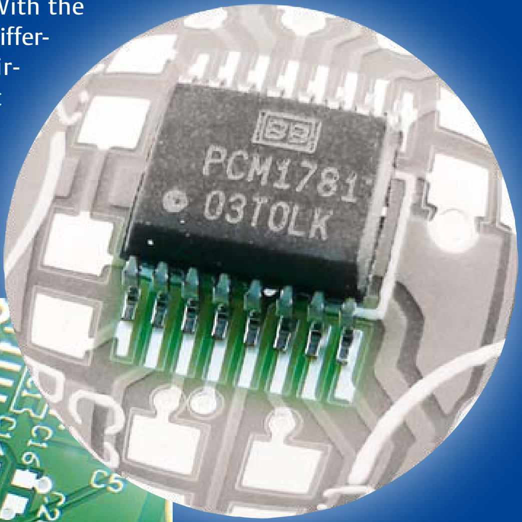
PCB design: beware the tiny details!

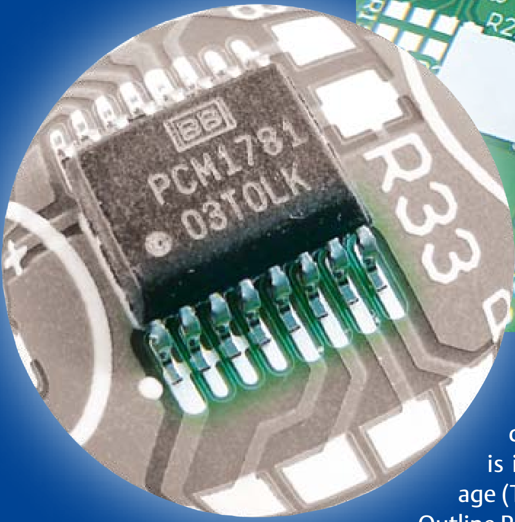
By Thijs Beckers (Elektor Netherlands Editorial)

You may have read the articles about our DSP system in the previous edition and the current one (if not, you should!). In the first two parts we described the features of a DSP and the programming software to be used with it. In the third part, which will appear in the September 2010 edition, we'll publish the schematic and describe the circuit board, amongst other things.

This circuit board contains a variety of SMD ICs in a number of different packages. The opamps used are packaged in a 'SOIC' type enclosure (Small Outline Integrated Circuit) with a pin spacing of 1.27 mm (0.05 inch), the DSP chip has a TQFP-package (Thin Quad Flat Pack) with 52 pins that are spaced with a distance of 0.65 mm (0.0256 inch) from each other,







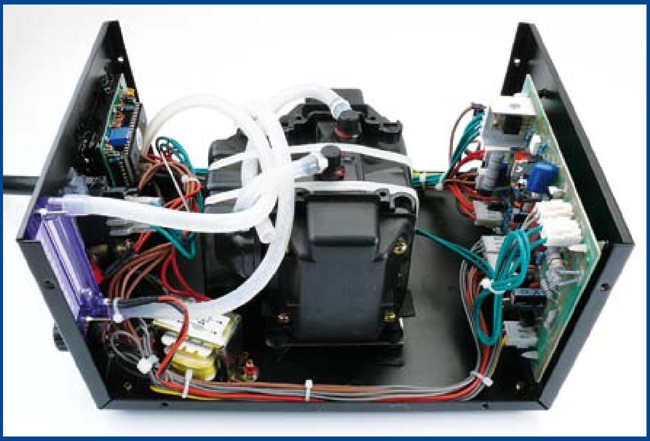
just as the A/D converter which is in a TSSOP package (Thin Shrink Small-Outline Package). With the sample-rate converter the spacing is another step smaller in a TQFP package with a 0.50 mm (0.02 inch) 'pitch' — the term used to denote the pin spacing. This was one of the reasons that we decided to offer this as a ready-built module; many people will have great difficulty working with these types of small components (they are unfortunately not available in bigger packages).

A small, tricky problem when designing the board was the package for the D/A converter. This chip, made by Texas Instruments, is housed in a SSOP/QSOP package (Shrink Small Outline Package/Quarter Size Small Outline Package) which looks a lot like the TSSOP package of the A/D converter, but this one has a pin pitch of 0.635 mm (0.025 inch). The difference is only 0.015 mm (15 thousandths of a millimeter)! With the naked eye this difference on the circuit board is not detectable (see 'overview photo'). So that we can nevertheless show you this minuscule difference we have put the D/A converter IC both in its correct place (left, with 0.635 mm pitch) and in the position of the A/D converter (right, with 0.65 mm pitch).

At high magnification we can see that the pins align with the solder pads perfectly only when the IC is in its correct place (see magnified image bottom left). But when we place it in the position for the A/D converter (see magnified image bottom right) then we can see that there is a difference of about half the width of a pin and that the IC would not be soldered nicely in this place. It would probably work, but it is not tidy and the risk of short circuits is considerably greater. Old hands at SMD board design will probably shrug their shoulders and say: "Everybody knows that?! You must always check the pin pitch and not rely solely on the package name", but forewarned, forearmed! And I have to honestly admit that I didn't know that here was a package variation with a pin pitch of 0.635 mm. We'll just say that we're never too old to learn...

(110394-I)

Hurricane at SMD scale



By Thijs Beckers (Elektor Labs)

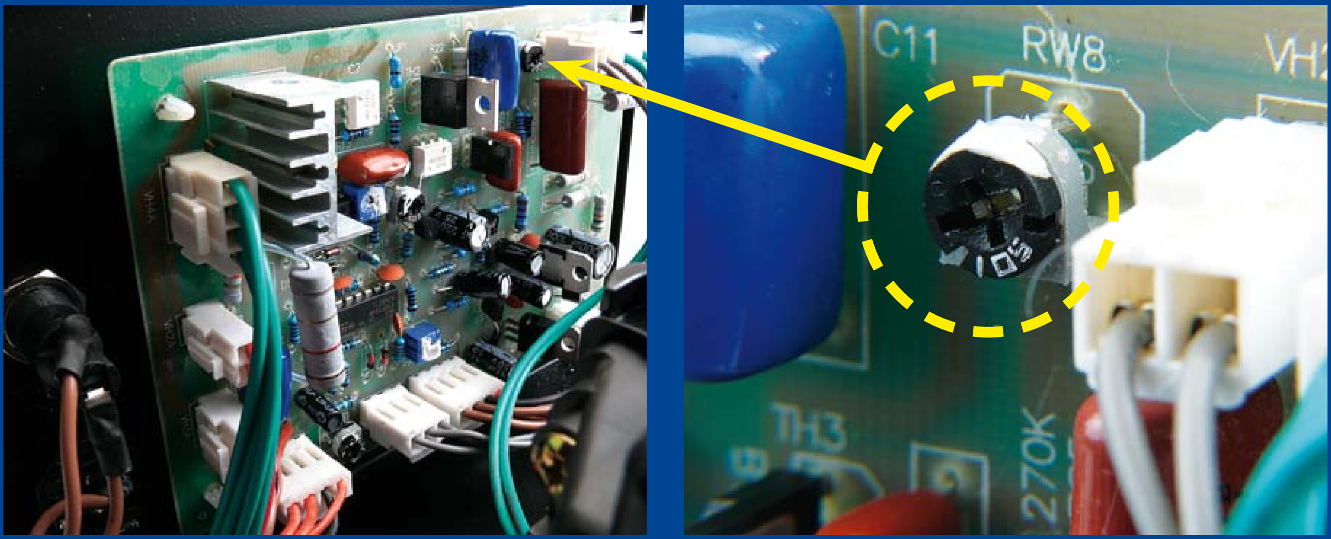
My fellow lab worker Luc Lemmens is a ‘convert’. He has been persuaded to buy a hot-air rework station for use at home. In our lab we have been using the Aoyue 852A+ to our satisfaction for a while now. I have to say, and I speak from my own experience, that soldering (and desoldering) of SMDs is really a lot easier, nicer and above all much tidier than with an ‘old-fashioned’ soldering station. Incidentally, desoldering SMD ICs is practically impossible using an ‘ordinary’ soldering iron without special attachments. With a hot-air rework station it is just a case of moving the nozzle along the pins of the IC until the solder melts. And that will happen, if you do it right, within five seconds.

Fair enough. Luc chose to buy the 852 from Aoyue (without the ‘A+’). A nice workstation for a reasonable amount of money, nothing wrong with it. Until you point the nozzle at a small SMD resistor... A category-5 hurricane pales by comparison. Resistor gone.

‘Well, then you just turn he airflow down a little?” Yes, if that was only possible... because the potentiometer for adjusting the airflow was already fully turned to its minimum position. No, this was thus not an ‘out of the box and work straight off’ experience. Now, Luc, as electronics engineer, considers it of course nonsense to immediately return the unit without first taking a look whether he can solve the problem himself. Grab a screwdriver and open that box!

The contents exceeded our expectations. For the price you pay you get a reasonably finished unit. A tidy circuit board, a decent compressor and a standard display. A grouch will notice the transformer which does not have a CE approval mark...

Looking at the circuit board it was obvious that there was a trimpot in series with the potentiometer for the airflow control (see photo). Would it be sufficient to give this a tweak? Sure enough, Luc was in luck. With this trimpot the airflow could be reduced much further than what was possible with the knob on the front panel only. After a bit of experimenting a more usable setting was found and the box could be closed up again. But before that, quickly swap the two front panel potentiometers, because you nearly always read the air flow and temperature settings



from the display and not from the rotary knobs. The potentiometers are positioned more logically when they are the other way around. The scale on the front panel is now no longer correct, of course, but that is not a problem when you're the only user of the station.
With a feeling of satisfaction the rework station is now really ready for use, without having to wait the number of days it would have taken if you had done it the 'official' way (return it). And then there would still be no guarantee that the new unit would be any better/usable.
However, beware that it is quite possible that the warranty after such an operation is likely to be void! Not that Luc has a problem with that, because if something goes wrong with it he will just fix it himself!

By the way, make sure that you do not turn the wrong trimpot! Because, in addition to the one mentioned here, there are a few more. These are for, among other things, the temperature setting. That got us thinking: does anyone know how you would calibrate the air temperature from the rework station? Where, for example, do you measure the temperature of the hot air? You can send your comments to editor@elektor.com and Jan will forward them.

(110261-I)

Finally, be careful when adjusting the air flow because the circuit is at AC power potential.

Down to Earth

By Thijs Beckers (Elektor Labs)

Recently my junior colleague Raymond Vermeulen was working on a circuit proposal received from a Belgian author for publication in the upcoming July/August double edition, also known as *Project Generator* edition. Although Raymond's French leaves much to be desired, it's a fortunate fact that reading electronics circuits does not tax language one's skills too much. And the circuit was very simple indeed.
It was in fact a detection circuit allowing the user to determine whether the earthing pin of an electrical outlet is properly connected. This is indicated by the circuit using a neon lamp that lights up when everything is all right. However, the circuit as proposed by the author seemed odd in one respect.
To check the operation of the circuit, Raymond built it up using an IEC appliance cord (see photo). A test in the Elektor lab seemed to confirm correct operation. However, when the plug was inserted in a few AC outlets outside of the lab, strangely the neon lamp seemed to light in some places and remain off in others.
This was quickly tracked down to the way the plug was inserted [1]. The circuit was obviously sensitive to the Phase (and Neutral). Except in the lab!
An e-mail the author failed to produce an explanation and he reaffirmed that the 'circuit' should work no matter how the plug is inserted into the socket...
We now know why the circuit works both ways when plugged into a lab AC outlet. The lab employs a central isolation transformer to provide power to all workbenches, making them safe for the designers. The Protective Earth (PE) pin of the AC sockets is however directly connected to the PE rod. You could say that both AC power wires are floating, i.e. there is no real 'Phase' and a 'Neutral', only a floating alternating voltage. The capacitive coupling between the conductors and Earth is enough to light the neon lamp.

Further 'study' of the circuit on hand shows that when plugged into a regular power outlet, it acts as a low-pass filter. Flipping the plug around creates a high-pass filter with a corner frequency well above the AC grid frequency. This explains the neon lamp lighting in one situation and remaining off in another.
In this way, we've actually combined a Phase detector and a Protective Earth checker in one simple circuit. The electrical situation at the author's home is still a mystery to us, since the neon light always lights. If you can help us, write to editor@elektor.com.
The circuit diagram is up our sleeve and you may look forward to seeing it revealed in the upcoming July/August 2011 edition. Specialists of electrical engineering may already have an idea of what it will look like.

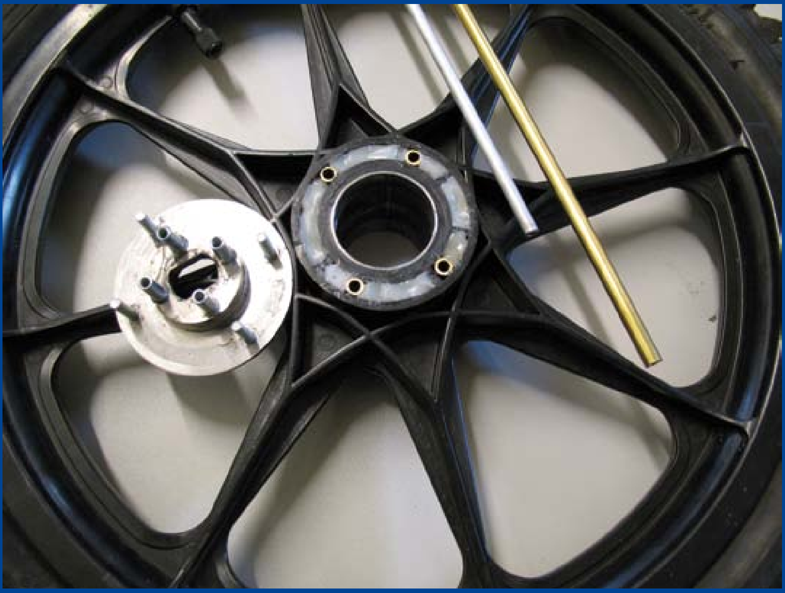
1. Editor's Note: Elektor Labs are based in the Netherlands. As opposed to the UK and the US, AC power outlets in the Netherlands are not polarized, i.e. AC power plugs (both earthed and non-earthed) can be inserted either way around.

Driver plate modification for ElektorWheelie

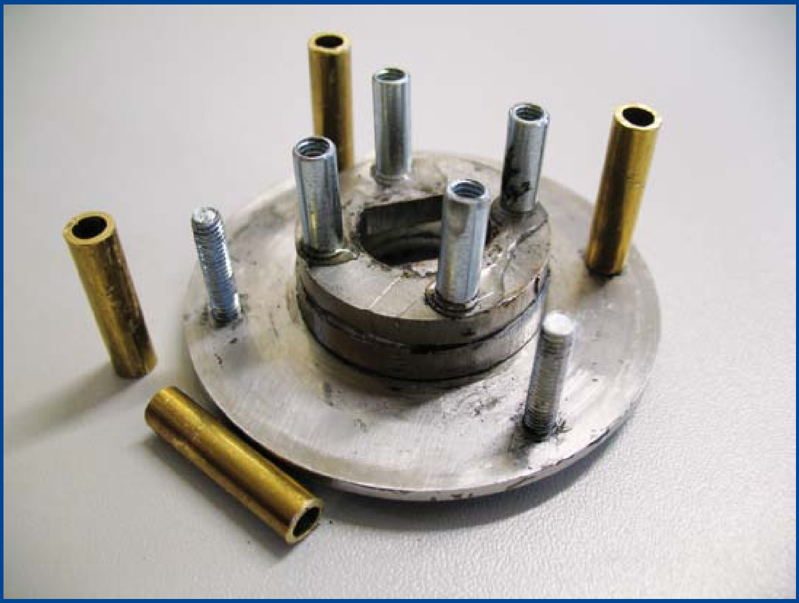
By Jan Visser (Elektor Labs)

As a fellow lab worker I am frequently informed about the technical problems that readers and lab colleagues are experiencing with projects. In this E-Labs Inside article I would like draw attention to one of these, which is relevant to the ElektorWheelie (see Elektor July/August and September 2009 and/or [1]).

After intensive use of the ElektorWheelie it appears that the carrying bolts (see **lower right photo**) can bend, warp or even break off either through intensive use or because of excessive oscillation. To prevent this we have come up with the following modification.



Take four brass or aluminum tubes with an internal diameter of 5 mm (0.2 inch) and an outside diameter of 6 mm (0.25 inch). These tubes can be cut from a longer tube and have to be 25 mm (1 inch) long. These will then fit exactly in the holes in synthetic wheels of the ElektorWheelie. Brass or aluminum tubes that could be used for this can be obtained from various model-building shops and perhaps also from building supply stores. Conrad Electronics [2] also has a suitable tube available. Search for part number 297321. The intention is for these brass or aluminum tubes to be glued into these holes (see **upper left photo**). Use two-component epoxy for this and apply it to the outside of the tubes. Use the wheel driver plate as a jig, so that the tubes will be in the correct place in the wheel and position the entire assembly into the holes in the wheel. Make sure that the bolts and driver plate are not glued to the wheel — you still have to be able to take it apart in a tidy way.



Allow sufficient time for the glue to set (overnight) and then remove the driver plate with the bolts from the wheel. The tubes are now held rigidly and in the correct place in the wheel. To make the entire assembly even more rigid, fill the remaining gaps in the holes in the plastic with more glue. You could use either two-component epoxy or hot-glue for this.
Once the glue has hardened sufficiently, you can reattach the wheels to the ElektorWheelie. Do not forget to use Loctite again on the center nut (which attaches the driver plate to the shaft) to prevent it from coming loose.

With this simple modification you reduce the forces on the wheel carrying bolts and the shocks they experience during oscillation. The wheel is picked up immediately and in this way you prevent the bolts from warping or even breaking off.

(110395-I)

Internet Links

[1] www.elektor.com/wheelie
[2] www.conrad.com

The Entire Circuit Cellar Archive Ready to Go Where You Go

Choose the Style that Fits You Best

CC Gold

A Stylish, Durable, Portable, Gold-Plated USB Drive

CC Key

The Key to Your Success Is Always Handy

CC Slim

The USB Archive that Fits in Your Wallet

The complete Circuit Cellar issue archive and article code stored on a USB flash drive. You can easily keep your CC Gold archive up to date by purchasing subsequent issues from our webshop or by downloading issues with a Circuit Cellar Digital 2.0 Subscription. Issues appear in searchable PDF format.

Complete archive includes PDFs of all issues in print through date of purchase.

Visit www.cc-webshop.com to purchase

Circuit Cellar Archive Index

Here comes the Bus! (6)
Elektor experimental PCBs and more

Our bus reaches the next stop on its route with the appearance of a couple of boards aimed at application development. These easy-to-assemble units include an experimental node with analog and digital inputs and a compact USB to RS-485 converter. We also describe a simple system that guarantees efficient and reliable communication on the bus.

By Jens Nickel (Elektor Germany editorial)

Spurred on by our encouraging initial results, we now proceed to describe a couple of simple boards that can be used to develop bus-based applications.

The basis of our experimental node is the circuit of the ‘test node’ we gave in the previous installment of this series. That design sported just a ‘test LED’ and a ‘test button’, which were enough to check whether messages could be sent successfully over the bus. All very worthwhile, but not enough to develop realistic applications.

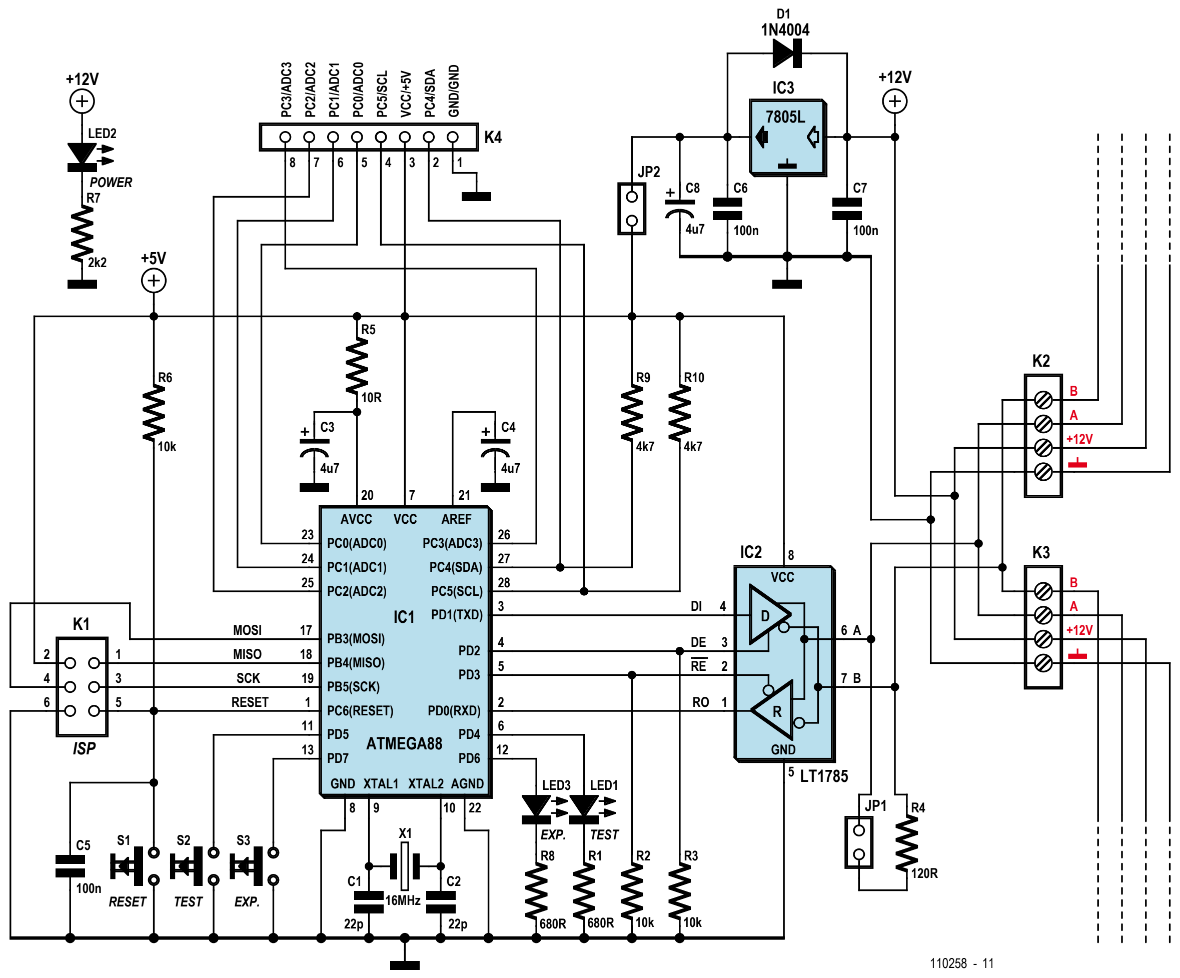
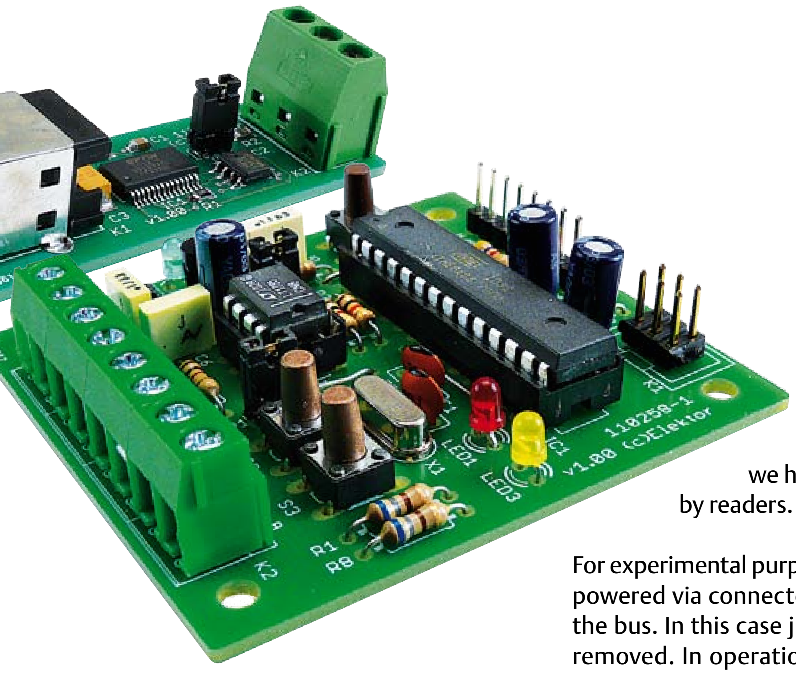


Figure 1. Circuit diagram of the experimental node.



Experimental node

The circuit diagram for the new ATmega-based bus node is shown in **Figure 1**. The operation of the RS-485 bus driver was described in the previous article in this series. The doubled-up connector blocks allow the node to be connected to the bus and for the bus signals to be wired on to the next node. A small extra feature is the jumpered connection to a 120 Ω resistor, which allows the RS-485 bus to be properly terminated at each end.

We have connected an extra LED and an extra button to port pins PD6 and PD7: we will call these the 'experimental LED' and the 'experimental button'. The most significant new feature, however, is header K4, where six additional microcontroller pins and the microcontroller power supply and ground connections are brought out. The decision to use port pins PC0 to PC5 was carefully considered, as these (as is the case in many microcontrollers) have various special features. All the pins can be used as digital inputs or outputs; PC0 to PC3 can alternatively be used to measure four separate analog voltages in the range from 0 V to 5 V with the help of the microcontroller's built-in multi-channel analog-to-digital converter; and PC4 (SDA) and PC5 (SCL) can be used to provide access to the I²C interface in the device. In accordance with the specification of that bus we have equipped the SDA and SCL signals with a pull-up resistor each.

It is therefore possible to connect simple sensors, I²C bus slaves such as temperature monitors and other similar devices to K4. The ATmega88 datasheet [1] shows how the pins are controlled using the registers in the device. Later in this series we will develop various applications using this connector, including, we hope, ideas suggested by readers.

For experimental purposes the node can be powered via connector K4 instead of over the bus. In this case jumper JP2 should be removed. In operation it is definitely recommended to connect the ground of the supply to bus ground as well as to earth. For more on this point, see the previous article in the series [2].
For completeness we should mention that we have wired the AVCC and AREF connections for the microcontroller's internal A/D converter as suggested in its datasheet.
The single-sided printed circuit board (**Figure 2**) is easy to populate. It is simplest to start with the wire links.

COMPONENT LIST

Experimental Node

Resistors
R1,R8 = 680Ω
R2,R3,R6 = 10kΩ
R4 = 120Ω
R7 = 2.2kΩ
R9,R10 = 4.7kΩ

Capacitors
C1,C2 = 22pF
C3,C4,C8 = 4.7µF
C5,C6,C7 = 100nF

Semiconductors
D1 = 1N4004
IC1 = ATmega88-20PU
IC2 = LT1785
IC3 = 78L05

Miscellaneous
X1 = 16MHz quartz crystal
S1,S2,S3 = pushbutton
JP1,JP2= jumper
LED1 = LED, 3mm, red
LED2 = LED, 3mm, green
LED3 = LED, 3mm, yellow
K1 = 6-pin (2x3) pinheader, lead pitch 2.54mm (0.1 in.)
K4 = 8-pin pinheader, lead pitch 2.54mm (0.1 in.)
K2,K3 = 4-way PCB screw terminal block, lead pitch 5.08mm (0.2 in.)
PCB # 110258-1 [3]

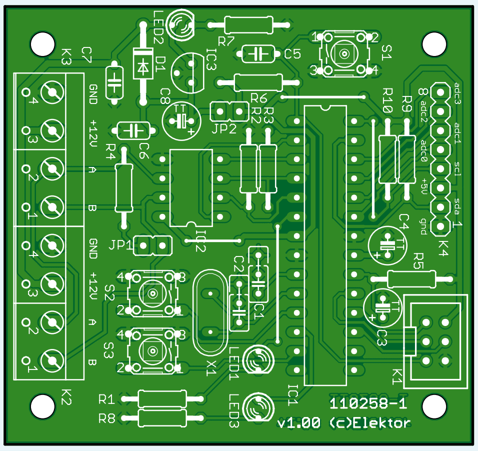
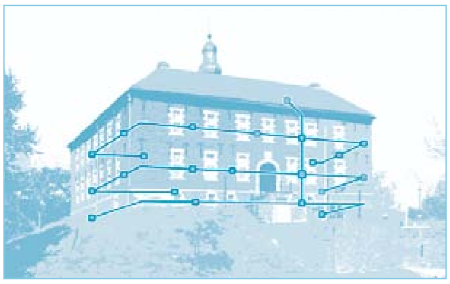


Figure 2. The circuit board is populated with conventional components throughout.



USB-to-RS-485 converter

There is relatively little to say about this board: it corresponds directly to the circuit diagram of the USB-to-RS-485 converter given in the previous article in this series (**Figure 3**). As the FTDI device is only available as a surface-mount component, we decided to use SMDs throughout the circuit and offer the board as a ready-assembled unit (**Figure 4**). I believe this is the simplest and most compact USB-to-RS-485 converter we have ever published in Elektor. It is ideal for developing applications based on the bus, for example with the help of the PC software described in the previous article. A minor caveat is that the design of the ElektorBus system is not yet finalized, and so we cannot at the moment say for certain whether we will be looking at a more intelligent connection between the RS-485 bus as the PC at some point in the future (more on this below). Meanwhile the converter is still a useful unit to have, and it can of course

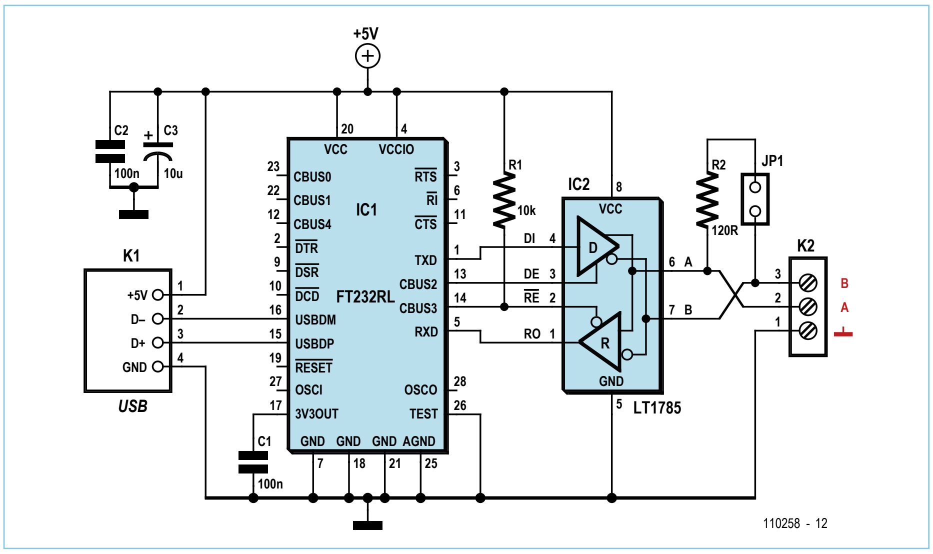


Figure 3. Circuit diagram of the USB-to-RS-485 converter.

also be used for other RS-485 applications. Since we are now set up with the boards we need for our experiments, we can turn to the question of software. The test software we described in part 5 [2] does not offer any mechanism to avoid (or even detect) collisions. Below we will look at a simple system which guarantees efficient and reliable communication on the bus. Alternative ideas have been discussed on our dedicated mailing list, and there is a brief summary of these in the text box.

Round-robin polling

In the previous articles in this series we looked at the pros and cons of cyclically polling the nodes. Each node in turn would be interrogated by a scheduler using a 16-byte message, and would reply with a message. A simple system like this would have two serious drawbacks.

1. A node that has a message that it wants to transmit must wait its turn. For example, consider what might happen when a user operates a light switch in a home automation system. In the worst case the node might have to wait while all the other nodes on the bus are interrogated. At 9600 baud with 32 nodes on the bus, it could easily take over a second before the light comes on, which is not acceptable.
2. There would be a large number of empty messages on the bus. It makes no sense, for example, to interrogate a light switch every second throughout the day if the switch is only operated perhaps ten times during that day.

The other extreme

In the case of nodes, such as switches, which only need to communicate infrequently, an alternative approach is better. The node can be arranged to transmit only when the switch is operated, without regard for the other nodes on the bus. Of course, this means that collisions can occur, resulting in random sequences of bits appearing on the bus. Instead of a neat 16-byte packet we suddenly receive a longer sequence of bytes with scrambled values, neither of the two colliding messages making it through unscathed. This means that, for important messages at least, the receiver should send an acknowledgement of safe receipt. If no acknowledgement is received by the original sender, it must repeat the transmission. This simple protocol also ensures that messages do not get lost as a result of other interference on the bus.
However, there are disadvantages. If several nodes are active simultaneously, all sending messages relatively frequently, then there will be many collisions. Also, a processing node that receives, for example, regular temperature readings from a sensor node must match each reading with an acknowledgement message, which increases both the load on the bus and the risk of collisions occurring.

A happy medium

Reader Jürgen Lange and I independently hit upon the same idea: let's see if we can get the best of both worlds! We can switch periodically between a polling mode (nodes speak only when spoken to) and the other extreme (where nodes can send messages at will without regard to the activity of other nodes).

COMPONENT LIST

USB/RS485 Converter

Resistors
R1 = 10kΩ (0805)
R2 = 120Ω (0805)

Capacitors
C1,C2 = 100nF
C3 = 10µF 16V (6032)

Semiconductors
IC1 = FT232RL
IC2 = LT1785 (SO-8)

Miscellaneous
JP1 = jumper
K1 = USB socket Type A
K2 = 3-way PCB screw terminal block, lead pitch 5.08mm (0.2 in.)
PCB # 110258-2 [3]
or
ready assembled and tested PCB # 110258-91 [3]

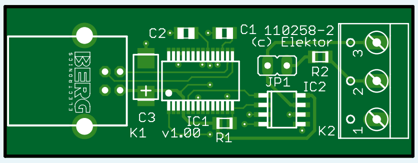


Figure 4. The USB-to-RS-485 converter is available from *Elektor* ready assembled and tested.

The alternative: a bit-level approach

When designing the experimental bus node, I thought that just developing the circuit would be enough of a challenge. However, that was nothing compared to the job of monitoring and leading the discussions on the mailing list. As alluded to in the previous article in the series, there was a number of experienced bus designers who were strongly in favour of a low-level approach to collision detection. CAN expert John Dammeyer suggested a method whereby each transmitter must wait for a 'space time' of 12 bit periods before sending a message. Since it also listens to the bus it can detect whether another node on the bus is talking. For various other approaches to collision detection that we discussed together, so-called 'bit banging' (direct manipulation of the UART port pins in software) would be required.

Other frequently-discussed topics were various low-level approaches to (re-)synchronization of bus nodes to the start of each message. *Elektor* reader Walter Trojan was in favour of using the 'nine-bit' mode of the UART, offered by the ATmega88 and various other microcontrollers. (Atmel call this 'multi-processor communication mode': see [1]). This sends each byte as nine bits rather than eight, the extra bit being used, for example, to distinguish between address and data bytes.

We cannot go into all the advantages and disadvantages of these various methods here. Suffice it to say that some of the more sophisticated approaches have what I see as a decisive disadvantage in that they restrict the choice of processors and platforms that can be used in bus nodes. Not all microcontrollers offer a nine-bit mode, and bit-banging in PC software strikes me (even assuming it is possible) as hardly elegant. And what if we want to send our messages over a different, perhaps wireless, network? The byte is a basic unit of information that practically any system can handle, whereas using nine-bit words and individual bits makes things rather more complicated.

As you might imagine, this discussion rapidly gathered pace. I earned a certain degree of opprobrium for wanting to keep open the possibility of controlling the RS-485 bus directly from the PC. Because of the timing idiosyncrasies of the Windows operating system this is quite a tricky proposition, and there is a lot to be said for replacing the USB-to-RS-485 converter with a USB-to-RS-485 gateway, which would include its own microcontroller to handle bus communications.

In summary: I felt that it was a decisive advantage to be able to transport our 16-byte messages seamlessly across different platforms and networks. In order to make the system maximally flexible, I decided that it would be best to ensure reliable and guaranteed communication as far as possible at the message level (see text) rather than at the bit level, closer to the hardware. Again, we welcome your feedback!

First, all the nodes that need to be interrogated periodically (such as temperature sensor nodes) are probed in turn. The scheduler then releases the bus for the unprompted transmission of messages. At this point any node that only occasionally has something to say (such as a light switch) is permitted to speak. The 'free bus phase' must of course only continue for a certain period of time, so that nothing is accidentally still being transmitted when polling mode resumes.

To implement this protocol, which I dubbed 'hybrid mode', I extended the software described in the previous installment [2]. First we will describe the implementation of the basic functions, which will subsequently be packaged up into a library; then we will look at a small application. Everything is, as usual, downloadable as source code from the *Elektor* website [3].

The scheduler

The PC takes on the role of the scheduler. Its transmitter address is defined as 0, which makes it easy for the other nodes to recognize its messages. The scheduler maintains an array, with *intPolledNodesMax* elements, containing the addresses of the nodes that are to by polled cyclically. It is also possible, of course, to arrange for a particularly loquacious node to be interrogated more often than the others.

To poll a node the scheduler sends out a special request message (*SchedulerRequest*), which includes the address of the polled node in the recipient address field. The scheduler then waits for a message with the same value in the transmitter address field (*ResponseMessage*), which can have any desired value in the recipient address field. The scheduler then turns to the next node in sequence and the process repeats. If a node fails to reply to a polling message, the process would come grinding to a halt. For this reason a timer is started when the *Sched-*

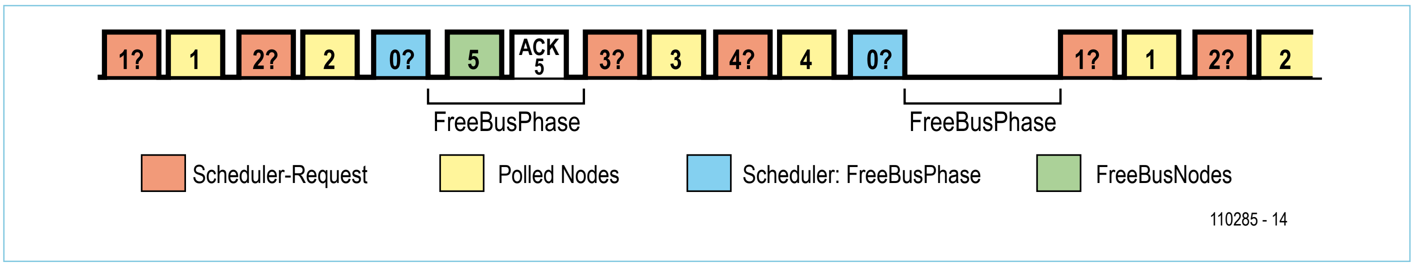


Figure 5. In h*ybrid mode* polling phases alternate with free bus phases. Collisions can occur during the *FreeBusPhase*, which means that the recipient must reply to each message with an acknowledgement.

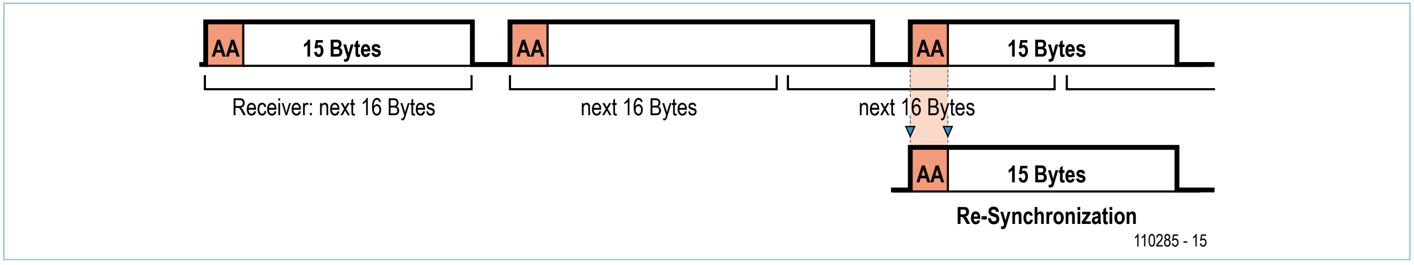


Figure 6. Our small application uses the first three bytes of the *Elektor* message protocol (the CRC is not implemented). Bit 0 of the mode byte indicates (by being set to zero rather than one) that the message originates from a polled node.

ulerRequest is sent out: if the timer expires without a reply being received, the scheduler stops waiting and moves on to the next polled node anyway.
The variable *intFreeBusCycle* has a special significance. It determines how many nodes will be polled cyclically before the scheduler switches to a *FreeBusPhase*. If the variable has the value 2, for example, this means that free transmission is permitted after every two nodes polled cyclically, as listed in the array (see **Figure 5**).
To initiate the *FreeBusPhase* the scheduler sends out a special *FreeBusMessage*, which has the recipient address field set to zero; the data payload bytes are not used. The scheduler then waits for a period *intFreeBusTime* (specified in milliseconds) before resuming polling.

Firmware

The BASCOM software from the previous article in this series was used as the basis for the code in the microcontroller nodes. Taking the advice of Günter Gerold on our bus mailing list, I modified the interrupt routine so that it returns immediately after receiving one byte. As before, the received bytes are stored in a byte array, and when the sixteenth byte is received, the routine checks wither the recipient address field matches the node's own address or is set to zero (a *FreeBusMessage*). In these cases the *ReceivedEventFlag* is set, which acts as a signal to the main loop that a message has been received and needs processing.
Now, when a node is directly interrogated by the scheduler (that is, using its own address), it is essential that it always reply immediately. The only nodes that reply to a *FreeBusMessage* are the nodes that are not polled (the *FreeBusNodes*), and even then, only when they have a message ready to send (when the *SendEventFlag* is set to true). In this case, if the message is to be sent out immediately upon receipt of the *FreeBusMessage*, then it is possible to avoid the use of a timer in the microcontroller to check for expiry of the *FreeBusTime* period.
A value (*PollingStatus*) is stored in the microcontroller's EEPROM to indicate the type of node. This allows the same firmware to be used in both node types.

When sending the message we make use of a hitherto unused bit in the mode byte (see the Elektor Message Protocol in **Figure 7**). Bit 0 being set to zero means that it is a *ResponseMessge*, and so collisions should not occur. If bit 0 is set to one then the message is part of the *FreeBusPhase* and so not safe from collisions: the receiver must reply with an acknowledge message. In no acknowledgement is received, then there has probably been a collision and the transmitter must send the message afresh, waiting at least until the next *FreeBusPhase*. To avoid repeated collisions, we need to ensure that the other message involved in the collision is not also resent in the same phase. A simple system ensures this: each transmitter waits for a different number of *FreeBusPhases* to pass before retrying. This number (the *FreeBusPriority*) is statically programmed into the EEPROM in the node, just like the node address. Further development of the software could allow the address, the *PollingStatus* and the *FreeBusPriority* to be changed dynamically.
In the course of a collision it is possible that sequences of bytes appear on the bus making packets of more than 16 bytes. This means that we need a mechanism to resynchronize to the beginning of the next message. In this version of the software we offer a partial solution to the resynchronization problem: we simply scan the bytes of the incoming data stream for the value AA_{hex}, which appears at the start of each message (**Figure 6**). The result of this approach is the restriction that this byte value may not appear within the data payload, and that we cannot use the CRC for error detection

Elektor Products & Services

- Printed circuit board for the experimental node, available from the Elektor Shop (110258-1)
- USB-to-RS-485 converter, available ready assembled and tested (110258-91)
- Free software download: PC software source code and BASCOM firmware

For software downloads and to order boards please visit www.elektor.com/110258 in case the byte value should appear there.

A small application

Now we turn to a small application, which will let us test our first experimental nodes. To produce a continuously-variable value that can be read regularly, I wired a 100 kΩ potentiometer to K4, between the 5 V, ADC0 and GND pins. I connected the experimental node to the bus with the two test nodes from the previous article in this series, which Günter had laid out and populated.

The experimental node was programmed with the address ‘02’ and a *PollingStatus* of ‘01’ in its EEPROM. The two test nodes were given addresses ‘01’ and ‘03’, a *PollingStatus* of ‘00’ and *FreeBusPriority* of ‘01’ and ‘02’.

In the interests of simplicity I used the same firmware for all three nodes. Pressing the test button sets the *SendEventFlag*, toggles the test LED and copies the LED status to the *LEDbyte*, which forms the first byte of the data payload of the message that will subsequently be sent (Figure 7). The value of *PollingStatus* also determines whether ADC0 will be read. (Remember that the ATmega88 needs to be told to use AREF as its voltage reference: see the source code.) We divide the ten bits of the ADC conversion result into two bytes, which will form the second and third data bytes of the message. Since we are not allowed to use the value AA_{hex} in the payload, we put the seven (rather than eight) least-significant bits of the result in the *ADClow* byte and the remaining three most-significant bits in the *ADChigh* byte.

We will use the PC to receive the messages, displaying the status of the three LEDs, as well as the ADC result, appropriately formatted. So as not to cause a conflict with the scheduler address (which is zero) we allocate a second address (10 in this case) to the PC. This address is used for sending acknowledge messages to nodes 1 and 3: in other words, the message has the recipient address set to the node address, the transmitter address set to 10, and the first data payload byte set to 16 plus the value of *LEDbyte*.

The secret is in the timing

After a couple of experiments it appeared that a value for *FreeBusTime* of between 50 ms and 70 ms was adequate. Since in this small application example the PC acts simultaneously as scheduler and as receiver for the microcontrollers’ messages, I rather inelegantly let it send an acknowledge message (asynchronously) just after the end of the *FreeBusPhase*, only after that returning to interrogating the nodes. Normally the scheduling and the sending of acknowledge messages should be done synchronously. The message containing the reading and the acknowledge message would then both fall within a *FreeBusPhase*, as shown in Figure 5.

It took several attempts before the software was running correctly. One of the bugs caused me a particularly large amount of head-scratching before I managed finally to track it down. The symptom was that the microcontroller firmware was losing occasional incoming messages. The explanation was that in the interrupt service routine I had not checked for the correct recipient address. When during the processing in the main loop (checking buttons and reading the ADC) more than two messages were sent on the bus, the second message was overwriting the variables required to process the first message properly. The fix was simply to check the recipient address in the interrupt service routine and then accept the message for further processing. Then more time is available for the application code in the microcontroller, such as for processing readings. Here we are helped by the fact that in hybrid mode we never send two messages in succession to the same receiver.

Outlook

Since the PC simultaneously takes on the personae of scheduler, bus participant and display unit, the current version of the software is somewhat untidy. In the future we will be able to simplify things by implementing the scheduler in a microcontroller. One possibility would be to use the microcontroller to control a (yet to be developed) USB to RS-485 gateway. This option has been suggested by various people on the mailing list: see the text box.

The resynchronization mechanism needs to be revisited so that we can allow the value AA_{hex} to appear within the data payload or CRC bytes. We already have some ideas for how to go about this.

Also, we want to look at how a node can announce its presence to the scheduler, and how to manage dynamic addressing. Last but not least we will also look in the next installment in this series at a realistic application. And, as ever, we invite you to participate and write in with your opinions and ideas.

(110258)

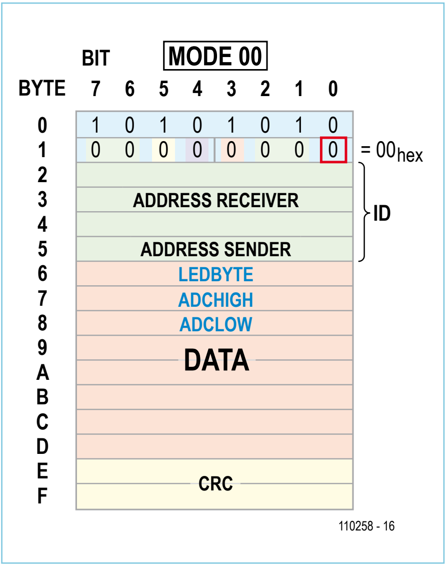


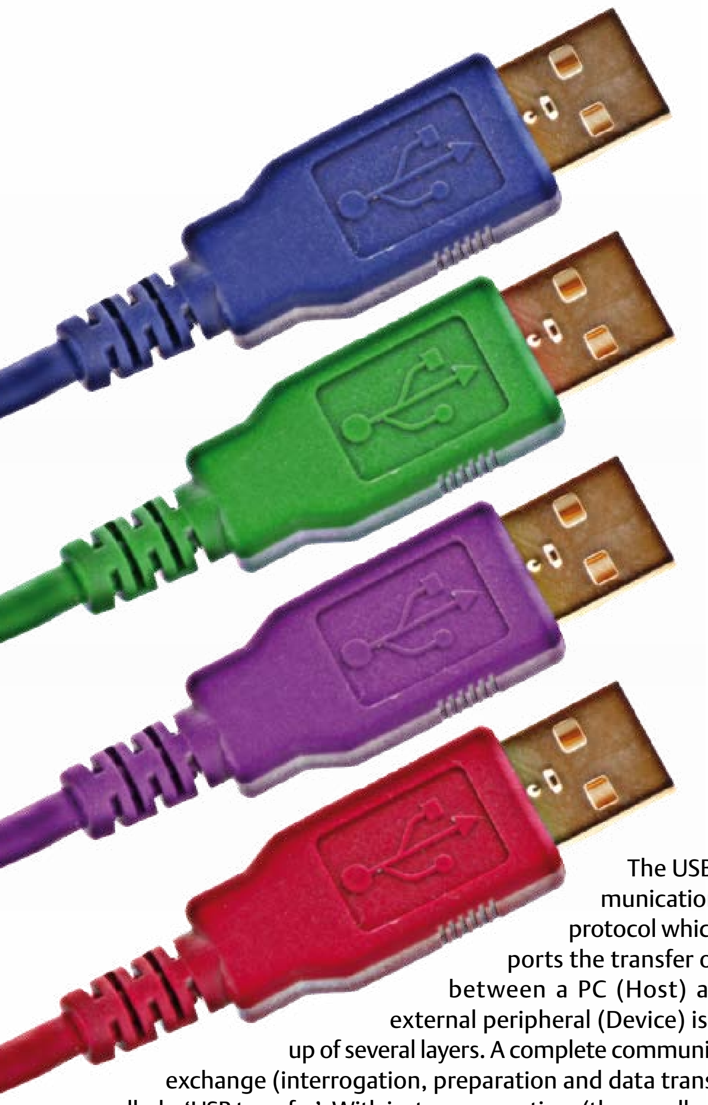
Figure 7. If a collision occurs sequences of more than 16 bytes can appear on the bus. In this case all the bus nodes need to be able to resynchronize to the start of the next message by scanning the stream for a byte with the value AA_{hex}.

Internet Links

[1] www.atmel.com/dyn/resources/prod_documents/doc2545.pdf

[2] www.elektor.com/110225

[3] www.elektor.com/110258



Inside USB

Around the turn of the century the USB interface elbowed aside the older serial and parallel ports to become de facto universal computer interface port. Consumers like its plug 'n' play convenience but as Elektor readers will know, it's the clever stuff going on under the surface which makes it all look so simple to the user...

By Guy Weiler (Luxembourg)

The USB communications protocol which supports the transfer of data between a PC (Host) and an external peripheral (Device) is made up of several layers. A complete communication exchange (interrogation, preparation and data transfer) is called a 'USB transfer'. With just one exception, (the so called 'Control-transfer') data transfer flows in one direction only.

A transfer consists of one or more transactions all of which must be carried out in a strict sequence. Each transaction is made up of a token packet (header), an optional data packet followed by a status packet (handshake). These packets are further subdivided, depending on the type of packet, they include the packet ID, the receiver address and error control information (see Figure 1).

Types of transfer

The Universal Serial Bus, as the name suggests is a general purpose interface port designed to support all types of data transfer occurring between an external device (peripheral) and a host PC. There are in total four transfer methods. Firstly the Control-transfer method is a special case and must be supported by every device (see below). The three other methods all handle the data flow differently to cater for different types of devices that will be attached to the bus:

- Bulk-transfer: where large amounts of data are transferred but do not require any guarantee of transfer speed or latency (e.g. from an external hard drive).
- Interrupt-transfer: For devices that need a guaranteed fast response but transfers a low volume of data (e.g. a USB keyboard).
- Isochronous-transfer: Occurs at a fixed transfer rate but without error detection (e.g. audio streaming).

With these last three methods the data always flows unidirectionally. These streams of data are composed of shorter transactions which are designed to ensure efficient and secure transfer between the device and host. 'IN' transactions pass data from the device to the PC whilst 'OUT' transactions transfer data from the PC to the device (IN/OUT is always seen from the PC's standpoint).
The transactions consist of three packets. The token packet specifies which type of transaction it is (In/Out). The data packet contains the data while the handshake packet gives feedback from the receiver. This can indicate either a successful transfer: (ACK) or 'I'm busy, try again later' (NAK).

Control-transfer

So far so good, it is a feature of the USB port that any supported device will communicate with the PC almost fully automatically on plug-in. The process whereby the host assigns an address to a device, reads information from it, loads the correct software driver, and then configures it is called enumeration. In addition it is likely that the device will not only communicate using one address but will have several different 'endpoints' to allow data to travel both to and from the device (see box). The host must therefore determine how many and also the length of the endpoints.
The complete enumeration process is performed using a special bidirectional type of transfer called a Control-transfer (always using endpoint 0 IN and endpoint 0 OUT of the device). As the name suggests this type of transfer allows the host to control the device. Control-transfer is structured in the same way as other transfer types using a sequence of transactions:

1. One setup transaction.
2. One or more data transactions (Optional).
3. One status transaction.

These consist of write and read control transfers. Whether or not the host sends data to the device or reads data from the device is determined during the setup transaction.

De-mystifying the protocol

Transactions

Like all transactions the setup transaction consists of a token, data and handshake packet. The PC sends the token and data packet and if the device receives the data successfully it replies with the handshake packet 'ACK' (**Figure 2**).

Every request starts with a 8-byte long Setup packet. The first byte *bmRequestType* determines the direction of the request (Bit7=0 indicates host to the device while Bit7=1 indicates the reverse direction) and also if this is a standard or vendor specific request.
The remaining 7 bytes: *bRequest* (1), *wValue* (2), *wIndex* (2) and *wLength* (2) complete the setup data packet.
Firstly we will look at the write transfer shown in **Figure 2**. After data has been received during the data transaction the device acknowledges successful receipt by sending an ACK or if the device is busy it will tell the PC to wait by issuing a NAK or cancel (STALL). Should an error be detected in the token or data packet then the packet is ignored. Although only one is shown, several data transactions can occur one after another.

The Status-transaction now completes the transfer and the device indicates that all went well. Now the roles are reversed; the device sends (status) data and the host replies with a handshake packet. When the transfer is successful the device sends a 'zero-length' packet to the PC and the PC replies with an ACK. When an error occurs at an endpoint, the device sends STALL. Should the device still be busy it sends a NAK.

With the read control transfer (**Figure 3**) the device can react in three different ways to the In-token from the host PC. The device can supply the data or if the device is busy it can send a NAK. When a failure occurs with the endpoint then a STALL is sent. Any error will cause the token to be discarded and not returned.
During the Status-Transaction the host PC responds with a zero-length packet to indicate the data has been successfully received. The device in turn replies with an ACK handshake packet.

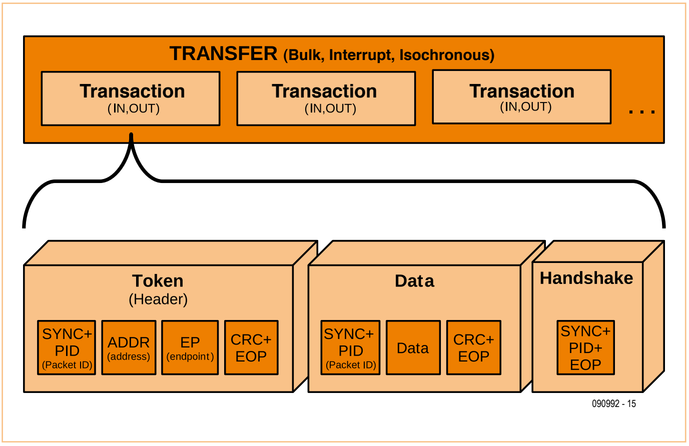


Figure 1. USB transfers comprise an inseparable sequence of transactions comprising a token, a data field and a handshake packet (Control transfer is a special case not shown here).

The descriptors

In order to discover which type of device has been connected the host issues Standard-requests to the device during the enumeration process prompting it to reply with its device descriptor information. This device description information is hierarchically organized [1]. Every device must supply a minimum of four descriptors:

- Device descriptor: Each device has just one device descriptor.
- Configuration descriptors: A device can have more than one configuration; for example it may be possible for it to be powered from the bus or from an external supply. There will only ever be one configuration active at any one time.
- Interface descriptors: This groups several endpoints into a single functional group. There can be a number of interfaces active

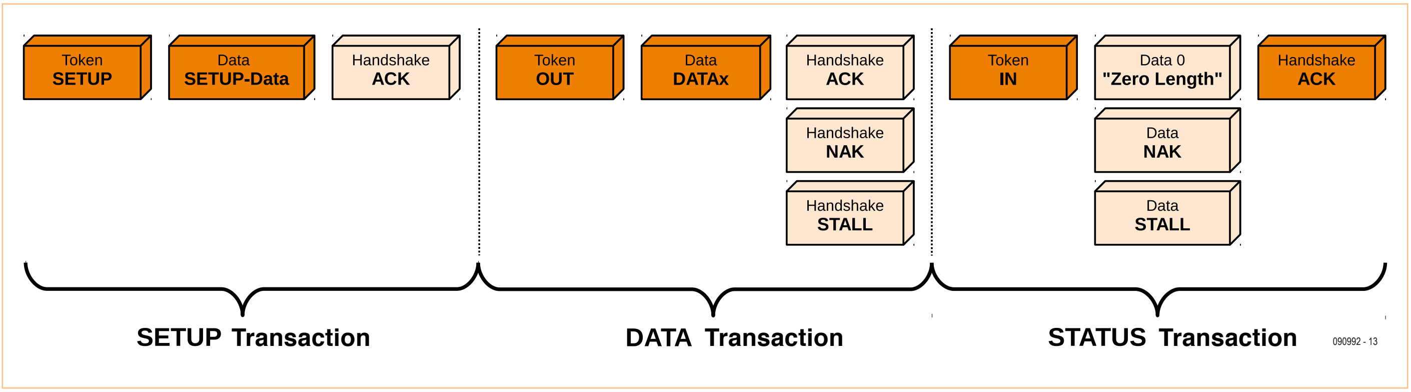


Figure 2. Each Control-transfer (in this case 'write') consists of a Setup-transaction followed by an optional (one or more) data transactions and a Status-transaction.

The device descriptor

Field	Lenght in bytes	Value in AVR-Library [3]	Meaning
bLength	1	18 (0x12)	Descriptor length in bytes
bDescriptorType	1	1	Device descriptor = 1 (constant)
bcdUSB	2	0x0110	USB_Spec1_1
bDeviceClass	1	0xFF	Class code (vendor specific, here = 0xFF, see [6])
bDeviceSubClass	1	0xFF	Sub-class code (vendor specific, here = 0xFF)
bDeviceProtocoll	1	0xFF	Protocol code (vendor specific, here = 0xFF)
bMaxPacketSize	1	8	maximum packet size for zero Endpoint (EP0_FS)
idVendor	2	0x03eb	Atmel Code assigned by USB.org
idProduct	2	0x0001	Product ID assigned by device manufacturer
bcdDevice	2	0x0001	Device release number
iManufacturer	1	1	Index of the manufacturers description string
iProduct	1	2	Index of the product description string
iSerialNumber	1	3	Index of the serial number string
bNumConfigurations	1	1	Number of possible configurations

The configuration descriptor

Field	Lenght in bytes	Value in AVR-Library [3]	Meaning
bLength	1	9	Descriptor length in bytes
bDescriptorType	1	2	Configuration descriptor = 0x02 (constant)
wTotalLength	2	39 (0x27)	Total length in bytes of the configuration descriptor including the interface and endpoint descriptors.
bNumInterfaces	1	1	The number of interfaces
bConfigurationValue	1	1	Number to use as a argument to select this configuration (must not be zero otherwise the device will be in a non-configured state)
iConfiguration	1	0	Index of the String-Descriptor which describes this configuration (0 = no text)
bmAttributes	1	0x80	Bit7 = 1 powered from the bus, Bit6 = 1 powered from an external source, Bit5 = 1 Remote Wakeup
bMaxPower	1	50	Maximum power drawn from bus in 2 mA increments

The interface descriptor

Field	Lenght in bytes	Value in AVR-Library [3]	Meaning
bLength	1	9	Descriptor length in bytes
bDescriptorType	1	4	Interface descriptor = 4 (constant)
bInterfaceNumber	1	0	Number of interface
bAlternateSetting	1	0	Value used to select an alternative setting
bNumEndpoints	1	3	Number of Endpoints in this interface excluding Endpoint 0
bInterfaceClass	1	0xFF	Class code (vendor specific, here = 0xFF)
bInterfaceSubClass	1	0xFF	Subclass code (vendor specific, here = 0xFF)
bInterfaceProtocol	1	0xFF	Protocol code (vendor specific, here = 0xFF)
iInterface	1	0	Index of String-Descriptor of this interface (0 = no text)

The endpoint descriptor

Field	Lenght in bytes	Value in AVR-Library [3]	Meaning
bLength	1	7	Descriptor length in bytes
bDescriptorType	1	5	Endpoint descriptor = 0x05 (constant)
bEndpointAddress	1	0x81	Bit7 = 1 (IN), Bit0 to 3 = Endpoint number (other bits = 0)
bmAttributes	1	2	Transfer type = Bulk (Contr. = 0, Iso. = 1, Int. = 3)
wMaxPacketSize	2	8	Endpoint FIFO size in Bytes
bInterval	1	0	Polling Interval = 0 (ignored for Bulk and Control), 1 for Iso, 1 to 255 for Interrupt

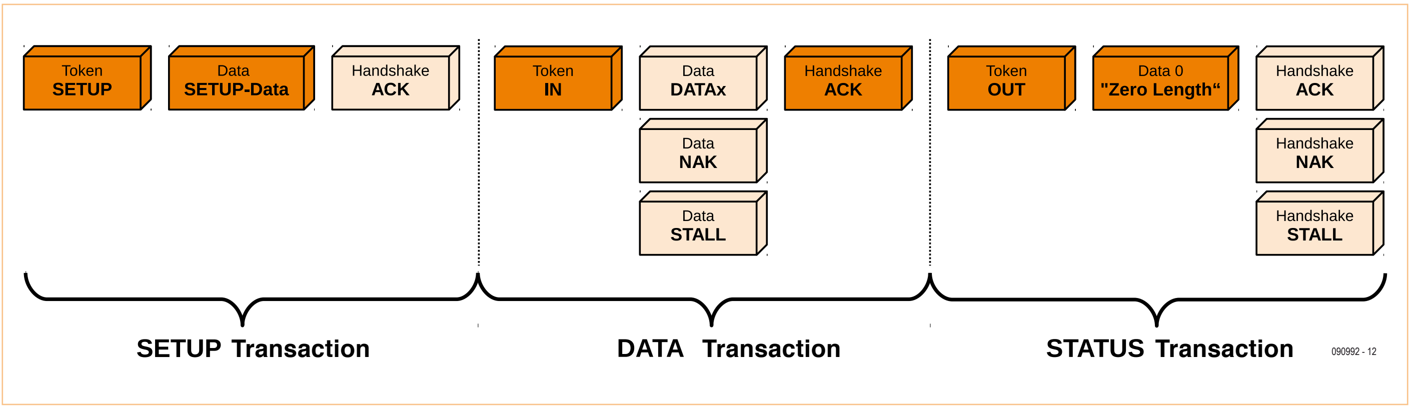


Figure 3. Read Control-transfer.

simultaneously (e.g. a Fax interface, a printer interface and a scanner interface all required for a multifunctional printer).

- Endpoint descriptor: These describe the endpoints (except endpoint zero). It is important to define direction (Bit 7) in the EndpointAddress field.

The String-Descriptors are optional and provide additional readable information. The strings are encoded in 16-bit Unicode. String index 0 gives a list of the supported languages.

Enumeration

This occurs when a USB device is connected; the host detects, identifies and then loads appropriate drivers for the device. For simplicity we will look at the process for one configuration and one interface. Once a device is connected to the bus the host can determine whether it is a low or high speed device by measuring the voltage level on both data lines. The host now pulls both data lines low to reset the device. Next it determines if the device is high speed or full speed by sending packets at full speed. Initially after reset the device communicates through endpoint 0 at address 0.
Now the (standard) requests will be executed using read and write control transfers. After a Get Descriptor request the host checks the eighth byte of the 64 byte device-descriptor to determine the maximum packet size that endpoint 0 supports (this can be 8, 16, 32 or 64 bytes). The host now issues a reset and assigns a unique address to the device using a Set address request.
The host requests 18 bytes containing the device-descriptor in the third transfer and the fourth transfer sends the 9 bytes of the configuration-descriptor. The data shows the total length of the configurations, interface and all Endpoint descriptors. A fifth transfer provides all these descriptors in one action.
The host PC now uses the vendor and product ID information to load the corresponding device driver.
The final enumeration transfer initializes and activates the user endpoints (endpoints 1-15) the device is now configured and its interface enabled.
Any requests from the host that are not supported in the device firmware will cause a STALL.

Further reading

The author [2], together with Jean-Claude Feltes (also an Elektor author) have developed a minimal USB library for the AVR controller family. This is available for download from [3], (documentation for the routines is written in German). The USB library *libusb* is used at the PC end of the USB connection [4].
For more information on the USB interface go to [5].

(090992)JN

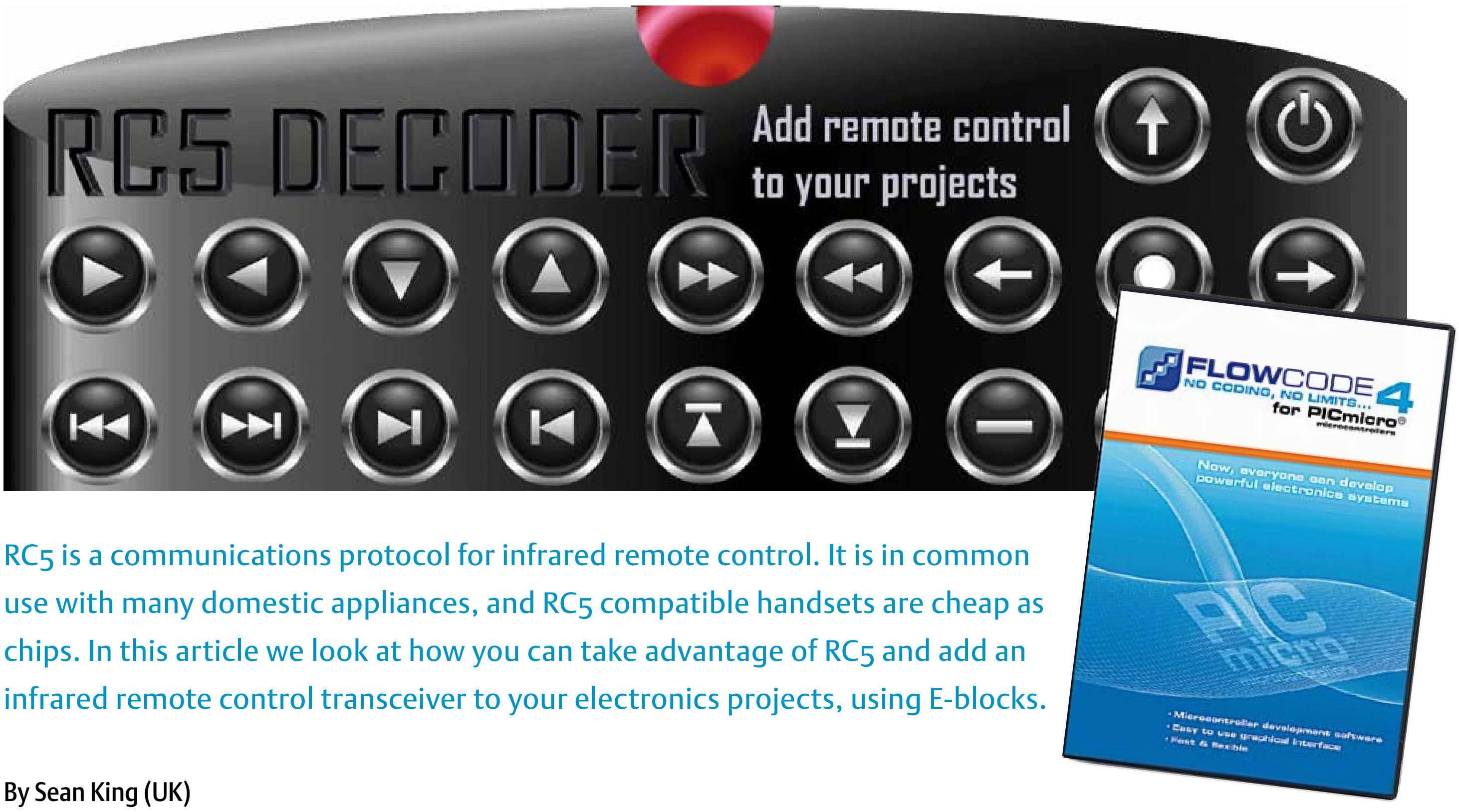
Weblinks

[1] www.beyondlogic.org/usbnutshell/usb5.shtml#DeviceDescriptors
[2] weigu@weigu.lu
[3] www.weigu.lu/b/usb/key/vendor/index.html
[4] http://sourceforge.net/projects/libusb/
[5] www.elektor.com/090768
[6] http://en.wikipedia.org/wiki/Universal_Serial_Bus

Endpoint

The data is transferred between the PC and the device *endpoint* (EP). The endpoint is some form of data storage device (FIFO, buffer, memory bank etc.) in the device and typically can store from 8 to 256 Bytes. Each device has multiple endpoints which can be addressed using the endpoint address (0-15). Bit 7 of this address defines the direction of the data communication (an OUT endpoint has Bit 7 = 0, an IN endpoint has Bit 7 = 1). Every device has a control endpoint 0 which is used, for example during the enumeration process. It is the only bidirectional endpoint. In a USB 1.1 device it consists of an 8 Byte FIFO memory. The number and size of the other endpoints will vary depending on the device. Each transaction is bound to one device and one endpoint address.

E-blocks: Flowcode RC5
Add remote control to your projects


RC5 DECODER
Add remote control to your projects
FLOWCODE 4
NO CODING, NO LIMITS...
for PICmicro®
microcontrollers
Now, everyone can develop powerful electronics systems
PIC micro
• Microcontroller development software
• Easy to use graphical interface
• Fast & flexible

RC5 is a communications protocol for infrared remote control. It is in common use with many domestic appliances, and RC5 compatible handsets are cheap as chips. In this article we look at how you can take advantage of RC5 and add an infrared remote control transceiver to your electronics projects, using E-blocks.

By Sean King (UK)

RC5 as an infrared remote control standard for televisions and consumer appliances has been around for so long that it has now become an unofficial standard for many applications. So how does RC5 work? In answering this question we need to look at the receiving hardware and the software required for a microcontroller to receive and decode the RC5 instructions. First let's look in some detail at the RC5 protocol which will give you an understanding of the hardware and software required to generate and decode signals.

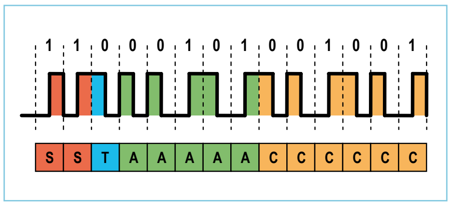


Figure 1. RC5 message structure. Start = 1 + 1. Toggle = 0. Address = 5 (typically VCR). Command = 9.

Protocol

RC5 is a relatively slow transmission standard. Each bit has a typical period of 1778 µs — with large variations tolerated. A normal RC5 transmission contains 14 bits of data (nearly 25 ms per message). The breakdown of a typical RC5 message is shown in **Figure 1** where it can be seen that there are 2 Start bits + 1 Toggle bit; + 5 Address bits + 6 Command bits. The function of each of these bits is explained below.

Start bits: two bits, both with the value '1', indicate the start of a message, and can be used as timing references for the following data.

Toggle bit: changes state each time a new key is pressed on a handset. Repeated messages with the same toggle bit value indicate that the handset button is being held down and messages are being transmitted on auto-repeat (typical of the volume control operation).

Address bits: five address bits are used to identify the intended recipient of the message. Some addresses have been allocated to common devices.

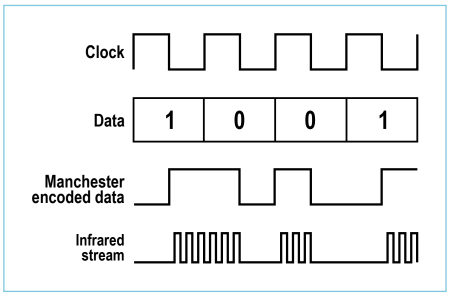


Figure 2. RC5 coding scheme.

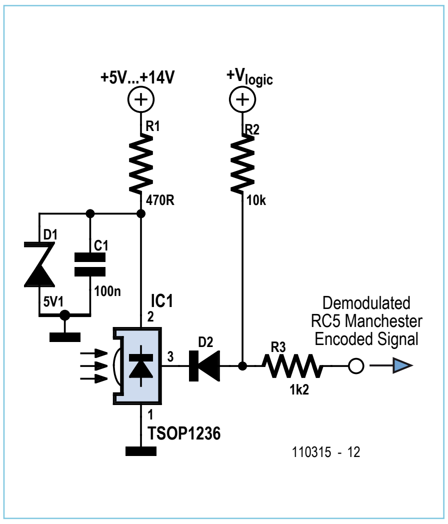


Figure 3 .The receiver circuit.

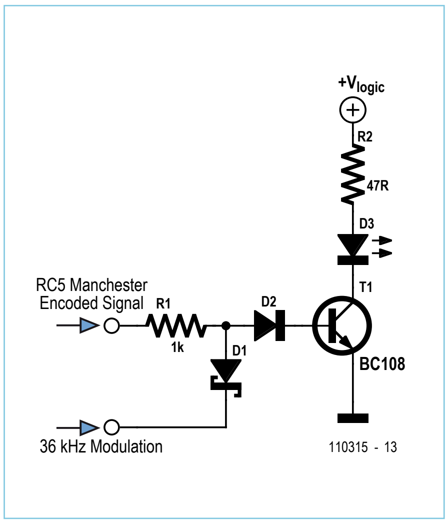


Figure 4. Transmitter circuit.

Primary TV	= 0
VCR	= 5
Satellite receiver	= 8

Command bits: six command bits indicate the operation to be performed. The numeric keys on a handset are usually transmitted as their numeric value. Other functions may vary, but tables of standard values can be found.

Encoding

When considering the communications link, the transmitter has a much easier job than the receiver. The transmitter dictates the signal timing and power, while the receiver must adapt to match the characteristics of any compatible transmitter and the effects of the transmission medium between. A fundamental problem with many communication systems is that the receiver needs to know when to sample the incoming signal — i.e. it needs to regenerate a clock for the data stream. To get round this headache, RC5 uses what is known as 'Manchester coding'. In Manchester coding each bit is represented by the direction of a transition in the middle of its time slot (see **Figure 2**). This guarantees that there will be at least one transition associated with each bit, helping to maintain timing synchronization in the receiver (albeit at the expense of increased bandwidth). The values of the preceding and succeeding bits determine whether there is also a transition at the start or end of the bit time slot.

Modulation

The final process before transmission of an RC5 signal is to apply 36 kHz modulation. This helps the receiver to remove ambient light pollution, set its automatic gain control, and ignore signals transmitted at other modulation frequencies.

RC5 hardware

Before examining the software, let's see what the hardware looks like. You can see the receiver circuit in **Figure 3**. This is quite straight forward: the receiver consists of an optical receiver module (TSOP1236) which is sensitive to infrared light. This receiver is actually quite a complicated beast that includes circuitry for removing the 36 kHz modulating signal and which also handles the tricky task of Automatic Gain Control for the infrared signal. But as far as our circuit is concerned, it behaves like an optically triggered transistor in which R1, D1 and C1 provide a 5 V supply. The demodulated signal is fed to a microcontroller input. This line is held high by R2. When an infrared pulse is received, the 'transistor' is switched on which drags the output down to 0V. **Figure 4** shows the transmitter: when the microcontroller sends out a logic 1, T1 is switched on and a pulse is sent out — note that the microcontroller has to generate the 36 kHz pulses, as we are using a straight forward infrared diode.

Development system

For the purposes of this project both the transmitter and receiver are contained on a new E-blocks board: The EB060 infrared board. This is a neat design — it includes

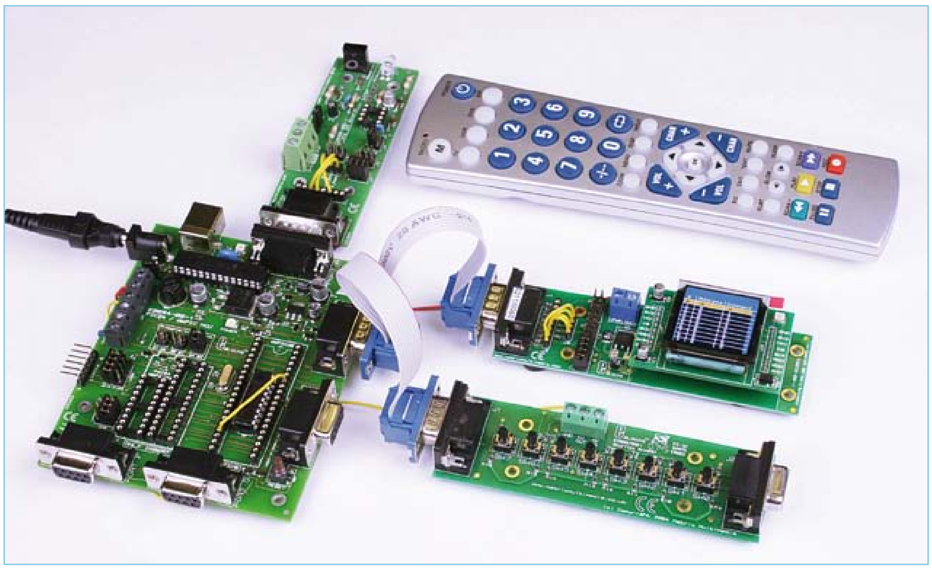


Figure 5. Test setup with E-blocks modules EB060, EB064, EB058 and EB007.

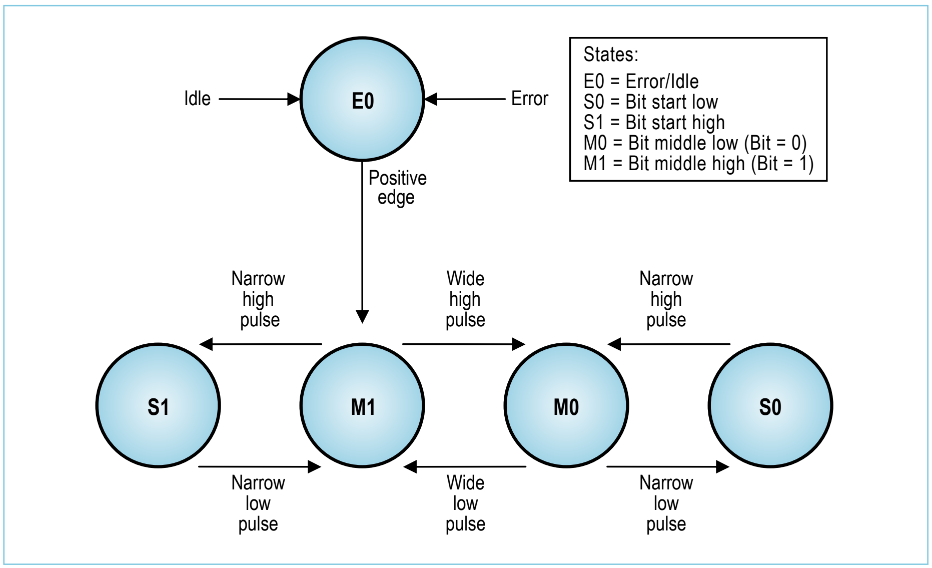


Figure 6. RC5 decoding state machine — states and transitions.

a socket for a separate PIC10F chip which you can use to set up as a modulator for the transmitter, freeing up your main microcontroller. You can see this set up in **Figure 5**. Here we are using a dsPIC Multiprogrammer board, a graphical LCD board and a switch board sharing port A, and an infrared board on port E. This dsPIC we are using is a dsPIC30F2011, which is an 18-pin device. You could use a conventional 8-bit PIC, but the dsPIC board is shiny and new so we are using that one. So now we have explained the hardware and the coding scheme, let's talk about the decoder — this is where the real work in developing such a system starts.

Decoding the signal

The RC5 system has been designed to be tolerant of significant timing variations. This was a requirement for early handsets with inaccurate oscillators and batteries operating over a range of charge states. The two start bits are always present in RC5 transmissions and can be used to synchronize the receiver's timer at the start of each message, allowing each data bit to be sampled at the appropriate time.

This is similar to the bit-banging approach often used with software RS232 receivers. However, slight errors in the measurement of a bit period will accumulate over several bits, which can lead to detection errors in the bits at the end of the message. Due to the guaranteed transition in the middle of each bit period, a Manchester coded message can be considered to consist of a combination of four types of pulses:

Table 1. Pulse width timings

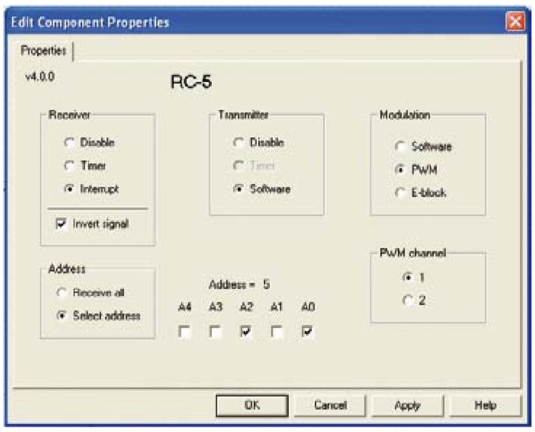
	Min.	Ideal	Max.
Wide (µs)	1334	1778	2222
Narrow (µs)	444	889	1333

1. a wide, high pulse;
2. a narrow, high pulse;
3. a wide, low pulse;
4. a narrow, low pulse.

These definitions can have large timing tolerances applied to them before there is confusion between a short and long pulse. This removes the need to know the exact transmission frequency, prevents errors accumulating with each bit detected, and allows decoding using software based on a state machine. So our first task is to define the times representing 'wide' and 'narrow' pulses by deriving from the RC5 ideal values within a standard tolerance of ±¼ bit period (444 µs). You can see these in **Table 1**.

Decoding state machine

In decoding the signal we need to have a technique that we can use to take in the infrared data stream and convert it into the simple numerical date we need. In practice we use a software trick called a 'state machine' to do this. You can see this represented diagrammatically in **Figure 6**. The state machine comprises five states. The arrows between the states indicate the available paths between states and the events that will cause the change to occur. The M0 and M1 states represent successful reception of a data bit (0 and 1 respectively). Each time



v4.0.0

RC-5

Receiver: ( ) Disable ( ) Timer (•) Interrupt

[✓] Invert signal

Transmitter: ( ) Disable ( ) Timer (•) Software

Modulation: ( ) Software (•) PWM ( ) E-block

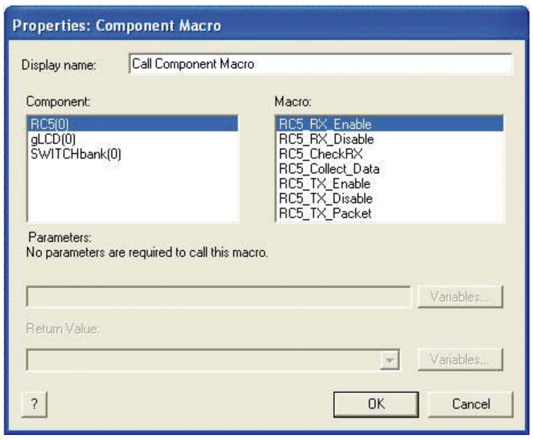
Address: ( ) Receive all (•) Select address

Address = 5

A4	A3	A2	A1	A0
☐	☐	✓	☐	✓

PWM channel: (•) 1 ( ) 2

Figure 7. Flowcode RC5 setup screen.



Display name: Call Component Macro

Component:
RC5(0)
gLCD(0)
SWITCHbank(0)

Macro:
RC5_RX_Enable
RC5_RX_Disable
RC5_CheckRX
RC5_Collect_Data
RC5_TX_Enable
RC5_TX_Disable
RC5_TX_Packet

Parameters:
No parameters are required to call this macro.

Figure 8. Flowcode RC5 setup screen.

the state machine enters one of these states, the corresponding bit value can be shifted into the receive shift register.

Error traps

If an event occurs that does not correspond with a path from the current state, the system generates an error and returns to the E0 state. In addition to illegal state transitions, the system will return to the E0 state if the measured pulse width is too short or too long to represent a valid signal

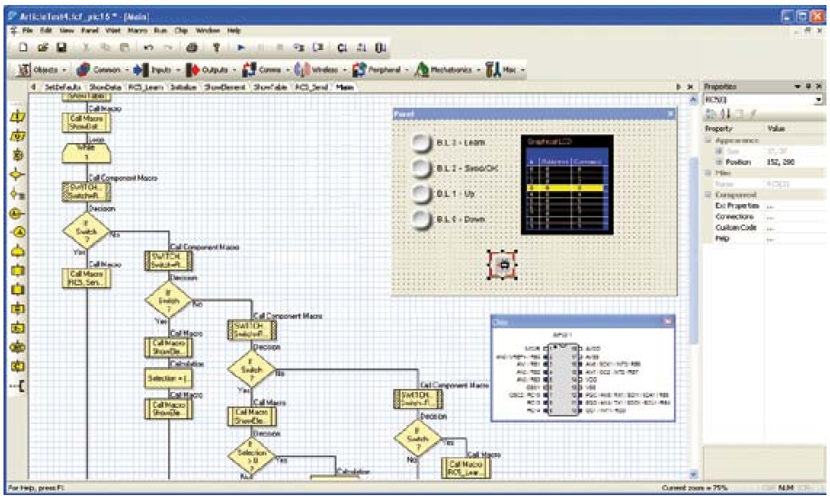


Figure 9. The Flowcode program.

Example using Figure 6

The receiver is idle and the state machine is in the E0 state. A positive edge is detected which moves the system to the M1 state and shifts a 1 into the receiver shift register. A narrow, high pulse moves the system to the S1 state. A narrow, low pulse moves the system back to the M1 state and shifts another 1 into the receiver shift register.
This completes detection of the two start bits and should always be the initial sequence of an RC5 message.
A wide, high pulse moves the system to the M0 state and shifts the Toggle bit into the receiver shift register as a 0. The remaining pulses move the system around the state machine until all the data bits have been recovered. In practice the state machine is implemented by setting the status of a number of variables in a program.

Implementation with Flowcode for dsPIC/PIC24

The final program was implemented in Flowcode for dsPIC/PIC24 which includes a new component macro designed to work with some of the variations of the RC5 standard that are commonly used. The Flowcode RC5 component allows a range of options for both transmission and reception of RC5 signals, including inversion of the received signal to compensate for the detector circuit, and optional filtering based on the incoming message address (see **Figures 7 and 8**). The macros provided allow complete messages to be created from their individual elements (Command, Address, Toggle) and transmitted using a single command, received messages to be detected, and the individual components of a received message to be read back. To help in debugging the program, we also used a graphical display E-blocks board. This was useful in showing the sequence of commands sent using infrared and the large number of characters allowed us to see the address and data parts of the RC5 signal to check our algorithm was working properly.
It's the clever work of many engineers that has allowed us to place our TV in the direct stream of infrared overload that sunshine provides, and still expect our remotes to work properly for less than a pound worth of components. To get round the issue of the high background levels of infrared radiation, engineers dictated that the RC5 standard should use a kind of pulse coded modulation. The infrared data consists of bursts of infrared at a frequency of 36 kHz. This is modulated by the microcontroller so that a logic 1 could be represented by a presence of an infrared burst and a logic 0 is represented by a lack of any infrared signal.

Flowcode program

The Flowcode program (see **Figure 9**) makes use of the RC5 component to develop a remote control transmitter that can learn and display codes from other hand-sets. The graphical LCD E-Block (EB058) displays a table of ten RC5 messages which defaults to address 0 and data values 0 to 9 (Main TV, buttons 0 to 9). The program is controlled by four push-buttons on a Switch E-Block (EB007) connected to the same port as the display (Port BL). The highlighted message is transmitted by the RC5 E-Block (EB060) when the Send button (Switch 2) is pressed. The yellow selection bar can be moved down and up using buttons 1 and 0 respectively. Learn mode is activated by pressing button 3. A menu screen is displayed, showing three options. Buttons 0 and 1 can be used to highlight the required option, with button 2 used to confirm the selection. If Learn mode is selected, the main table is displayed with the selection bar shown in red.
The selected message details will be overwritten by any valid message received by the RC5 E-Block, turning the selection bar back to yellow and restoring normal operation. Learn mode can also be cancelled by pressing button 2. The wire link shown on the U1 socket routes the RD0/INT1 pin to the PORTA/Miscellaneous connector where it is used for the receive interrupt. RC13 and RC14 are also available on this connector and are used to control transmission and modulation. Whilst this program has been written as to learn and regurgitate remote control commands, you should be able to modify it to perform any function you need. The Flowcode program can be downloaded freely from the Elektor website at [1].

Conclusion

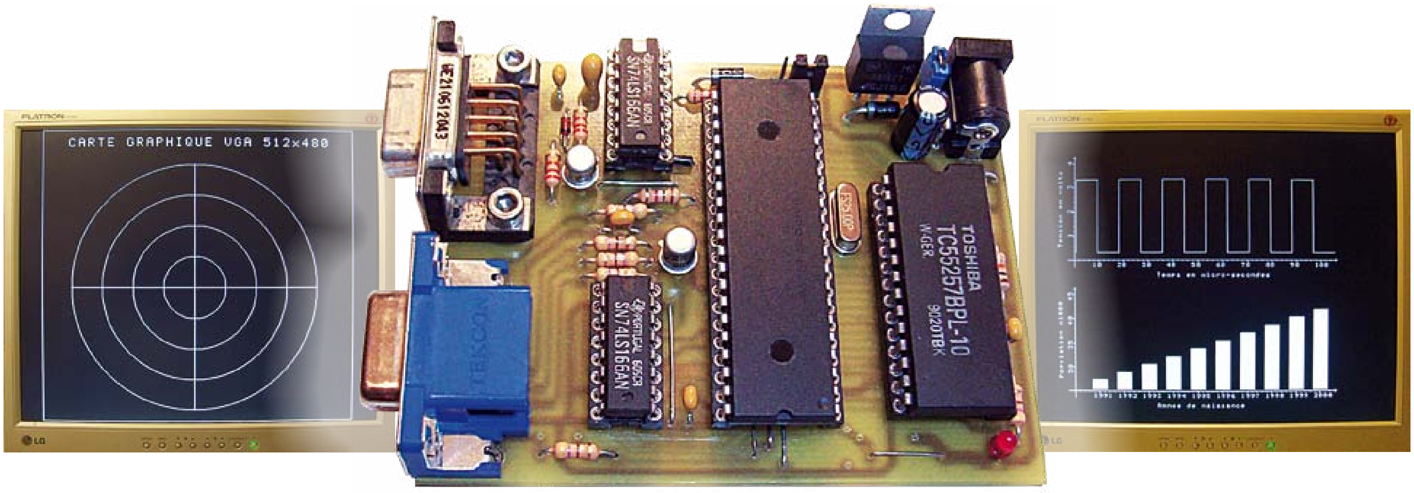
Encoding and decoding RC5 infrared signals can not be described as a 'trivial task', but in this article we have shown that the use of a state machine within Flowcode can considerably simplify the task of writing the program.

(110315)

Internet link:

[1] www.elektor.com/110315

VGA Add-on for Microcontrollers



Don’t bin your old monitors any more!

Many projects require a large amount of information to be displayed, but the size of the display itself is often a problem. One solution is to use an old 14” or 15” computer monitor that’s been scrapped but is still working. The VGA board described in this article lets you do just this, and is compatible with any microcontroller that has a serial port.

By Etienne Migot (France)

Microcontroller system designers usually limit themselves to a conventional alphanumeric display, up to four lines of 40 characters, but without the possibility of graphics — and above two lines, these get quite expensive. A graphics display, VGA monochrome or color, is dearer still and is more difficult to implement. It’s also possible to use a computer as a display system.
This last solution is used so as to have a convenient display, but suffers from several drawbacks, especially in that it monopolizes the computer when the display needs to be used continuously, or the difficulty of getting hold of ‘just the right’ software (e.g. for displaying graphs in real time).
The idea for this project came out of a simple observation: the increasing improvements in computer monitor technical specifications means that older models are almost automatically being scrapped. This applies to virtually all 14” screens and increasingly now to 15” screens, despite their more than respectable performance. Well, don’t throw those old screens away any more! We’re suggesting recycling them as top-quality display devices, by building the VGA add-on board proposed in this article. It can be interfaced with any microcontroller via a serial link, it offers simple, high-level command-based implementation, and is easy to build, using as it does only conventional components.

Overall architecture

The RS-232 / VGA converter architecture is based around a DSP microcontroller from Microchip (dsPIC30F3011), associated with a standard 32 kB RAM memory to hold the pixels displayed on the screen. The block diagram in **Figure 1** shows the interconnections between the various functions:

- Addresses A0–A14: read/write addressing for the 32 KB of RAM;
- Data D0–D7: data to be displayed in 8 bits (1 bit = 1 pixel);
- WE and OE are used to write or read the data in the RAM;
- SCK clocks the pixels out of the FIFO at 25 MHz;
- Hsync and Vsync are the Line and Frame synchronizing signals;
- RX-RS232 is the serial data input via which the characters to be displayed arrive, along with the commands in text and graphics mode.

The values of the crystal and the dsPIC’s internal PLL multiplier are such that it is

Note. Readers’ Projects are reproduced based on information supplied by the author(s) only.
The use of Elektor style schematics and other illustrations in this article does not imply the project having passed Elektor Labs for replication to verify claimed operation.

clocked internally at 100 MHz, i.e. a cycle time of 40 ns (four periods per cycle). The video signal coming out of the FIFO and the synchronizing signals are connected directly to the VGA output connector, the level being adjusted by the use of simple resistors that form a voltage divider with the monitor's input resistors.
The available resolution is of VGA monochrome type, limited to 512 × 480 by the 32 KB of RAM. To go up to 640 × 480, this would have to have a minimum size of 38.4 KB (640 × 480 / 8). Full VGA resolution would also have required an additional input/output for connecting the address A15, which is not possible, as the dsPIC has no more free input/outputs. Hence this choice would have led to a larger package not available in a DIL version, with all the associated mounting complications. For the same reasons, the monochrome version was preferred over a colour version, which would have needed at least three times as much RAM and extra pins on the microcontroller.

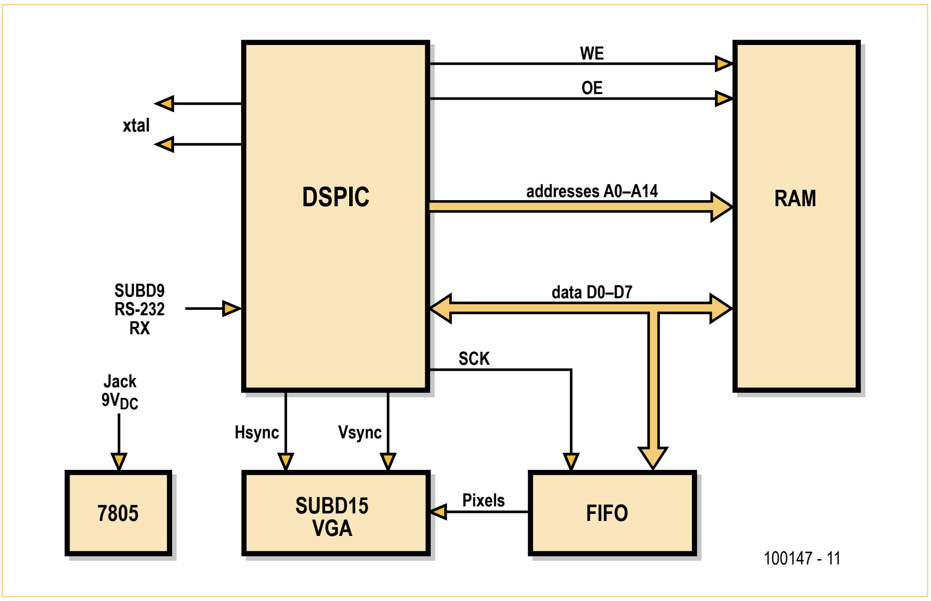


Figure 1. Overall architecture of the VGA Add-on for Microcontrollers.

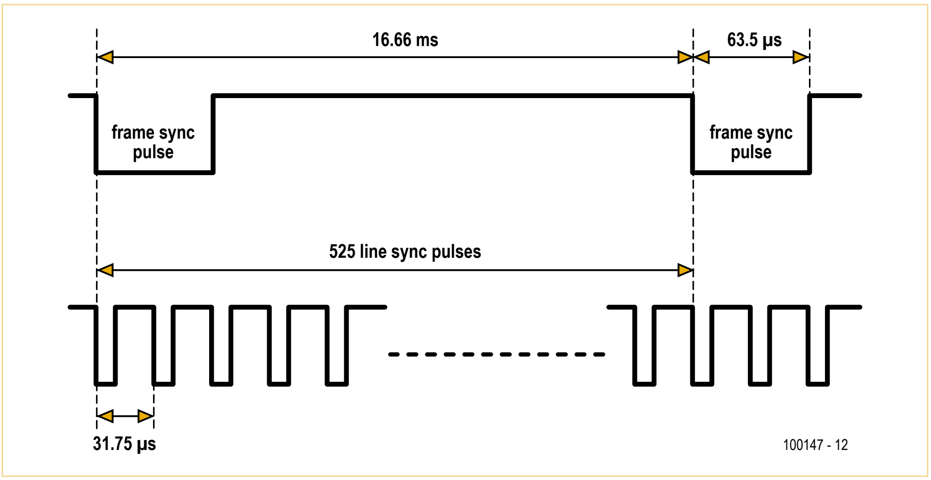


Figure 2. Timing diagram for a frame.

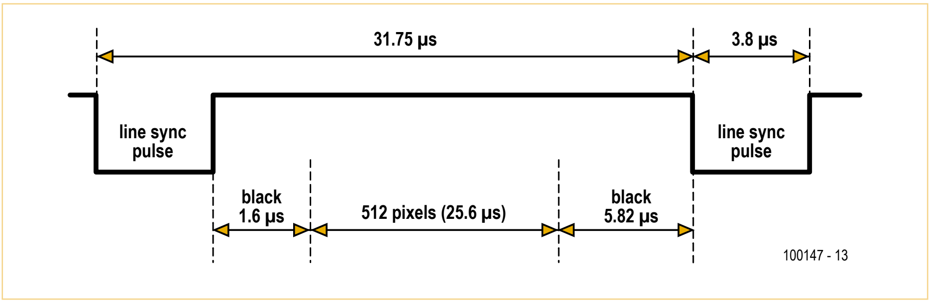


Figure 3. Timing diagram for a line.

Video signal

The screen display is a succession of images (frames), each image being made up of a succession of lines, themselves containing a succession of pixels. To keep it simple, the video signal may be considered as the combination of three main components:

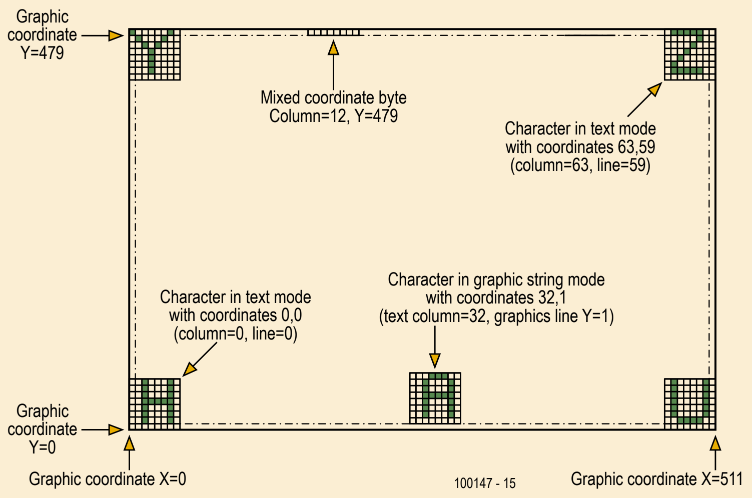
- a frame synchronization signal, marking the instant at which the scan recommences at the top of the screen;
- a line synchronization signal, marking the point for moving on to the start of the next line;
- the levels of the pixels to be displayed, the only information visible on the screen.

The synchronization signals are active low. For the pixel display, a low level corresponds to black.
The frame synchronization signal defines the time between the start of one image at the top left of the screen and the end at the bottom right. Refreshing usually takes place at 60 Hz, so the frame sync recurs every 16.67 ms. The pulse itself lasts for two lines. The frame is made up of 525 lines with, in order of appearance on the screen (**Figure 2**):

- frame sync lasting for two lines;
- 32 black lines;
- 480 lines containing the pixels, which make up the only part visible on the screen;
- 11 black lines.

Text and graphics co-ordinates

The board supports text (60 lines of 64 8-bit characters) and graphics (512 × 480 dots) modes simultaneously. The co-ordinate (0,0) is at the bottom left of the screen in both graphics and text mode. In text mode the co-ordinates go from (0,0) to (63,59), in graphics mode from (0,0) to (511,479). Certain commands support text-type co-ordinates in horizontal, and graphics-type co-ordinates in vertical, i.e. from (0,0) to (63,479).


Graphic coordinate Y=479
Mixed coordinate byte Column=12, Y=479
Character in text mode with coordinates 63,59 (column=63, line=59)
Character in text mode with coordinates 0,0 (column=0, line=0)
Character in graphic string mode with coordinates 32,1 (text column=32, graphics line Y=1)
Graphic coordinate Y=0
Graphic coordinate X=0
100147 - 15
Graphic coordinate X=511

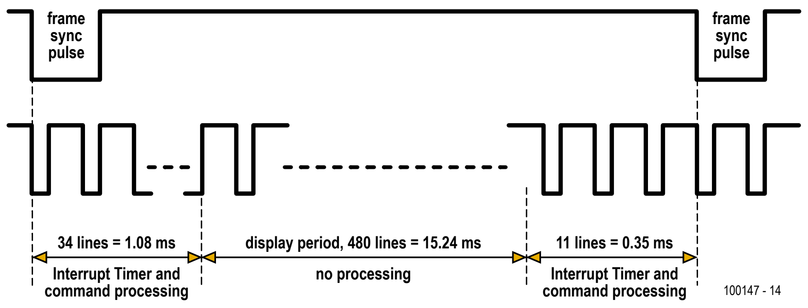

frame sync pulse
frame sync pulse
34 lines = 1.08 ms
Interrupt Timer and command processing
display period, 480 lines = 15.24 ms
no processing
11 lines = 0.35 ms
Interrupt Timer and command processing
100147 - 14

Figure 4. Processing / display time sharing.

The line synchronization signal defines the time between the start of one line on the left and the end on the right of the screen. The line sync recurs every 31.75 µs (16.66 ms/525) and lasts for 3.8 µs.

Each line (**Figure 3**) is made up as follows:

- a 3.8 µs line sync pulse;
- a black period before the pixels of around 1.6 µs;
- 640 pixels of 40 ns each, i.e. 25.6 µs;
- a period of black after the pixels of around 0.7 µs.

Monitors are fairly tolerant over sync pulses, the duration of the syncs can vary by 1 % without any adverse consequences. Within one line, the distribution of the time between the black areas and the pixels is even more tolerant. However, the instant of the appearance of the first pixel with respect to the line sync pulse must be extremely accurate, if we don't want to see wavy verticals on the screen.

In our case, only 512 pixels are displayed out of the 640 offered by the VGA standard, which means that 128 pixels will be 'black'. These are placed at the end of the line, so the black period after the pixels will be 5.82 µs (0.7 µs + 128 × 40 ns). Thus the visible part of the screen is not centered, but appears offset towards the left.

Software architecture

The software part is divided into three sections:

- Generating the synchronizing signals and reading the RS-232 input;
- Displaying the pixels (visible video part);
- Decoding the commands and processing them.

The generation of the sync pulses and of the pixel display must be precise so that the screen display will be perfectly stable. This is achieved entirely under the control of an interrupt timer (IT), balanced with much use of by NOP instructions, and occupies over 90 % of CPU time.

Decoding the commands, the graphics calculations, and writing the pixels to RAM has to make do with the remaining 10 % of CPU time (**Figure 4**). This is where all the power of the DSP comes into play, not just in terms of its operating speed, but also through the use of the internal DSP engine for the mathematical calculations (32-bit multiplication and division).
In certain cases, the calculations are so time-consuming that a cheat has had to be employed: "frame stealing", which involves not displaying the pixels for the whole duration of one frame, and thus having 100 % of the CPU time for 16.66 ms. However, we haven't overused this technique, as the eye does perceive the absence of one frame on the screen, creating an unpleasant visual sensation. So we've confined its use to commands filling the screen in 2D, which are the most time-consuming.

The serial port RX input is read during each line scan, which allows a high transmission speed. The value programmed at power up is 9,600 baud. You are then free to set the speed that suits you, up to 115 kbaud, by a dedicated command. A 512-bit FIFO software buffer makes it possible to stack characters and commands waiting to be processed, which can only be done outside the display period.

The software FIFO buffer is processed once per frame, the instructions received are processed in the order in which they arrive. Thus it is possible to send several tens of characters and/or commands to the buffer. The limit lies in the DSP's processing power and the time allocated to it for processing

all these commands. Because of the shortage of input/outputs, no provision has been made for the DSP to send information back to the serial port. So command stacking is to be used with care to avoid saturating the DSP — all the more so since there is no safeguard for the receive FIFO buffer.
The processing of a command begins after the display of the last pixel on a line, and ends at the first pixel of the next line. It can be split over several frames if the processing time exceeds the equivalent of 45 lines (1.43 ms), which is indeed the case for graphics commands in particular.
The software is available from [1].

Circuit diagram

The circuit diagram (**Figure 5**) is the direct realization of the block diagram. There's an LED to tell you the +5 V rail is present. The RS-232 RX input is normally at +5 V. It is driven by either 0 V/5 V signals on connector K5 (connection to a microcontroller), or by ±12 V signals on connector K3 (connection to a computer). The +5 V supply can come directly from your application (in which case jumper K2 should not be fitted) via K5 or via K3 (pin 9 of K3 has been set aside to allow you to feed the +5 V from your own board). Otherwise, you'll need to use an external +9 VDC PSU.

PCB

The PCB can be produced single-sided with the use of wire jumpers, or double-sided with a ground plane. The tracks are fairly broad, so there shouldn't be any real difficulties here. The signals being carried on this board are logic signals at 25 MHz, so careful construction is important. With a single-sided board, the occasional spurious artefact may appear on the screen, but the PCB is easier to make. With a double-sided board, the copper groundplane on the component side must be connected to a single ground reference point (pin 8 of IC4, for example).

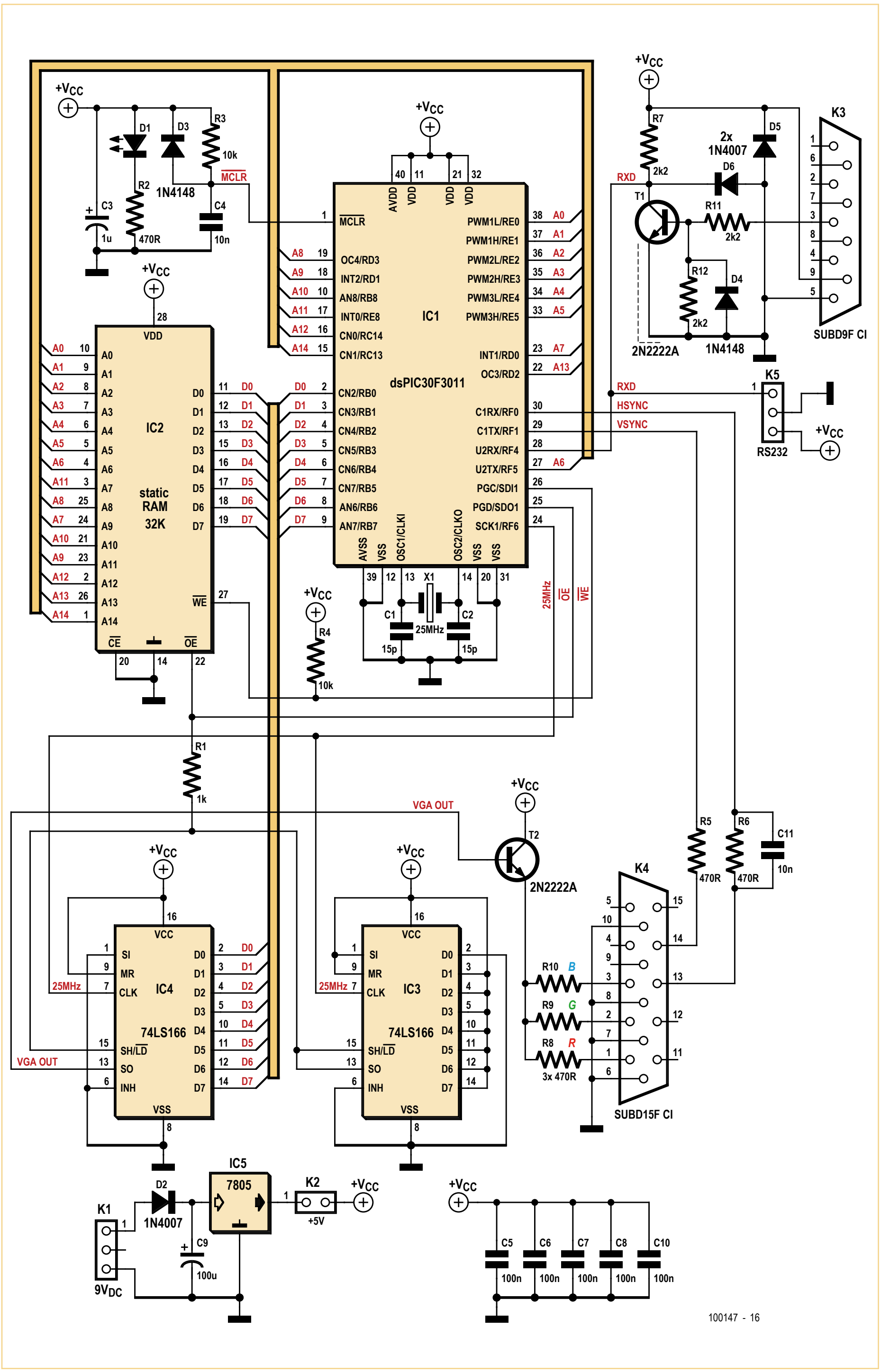
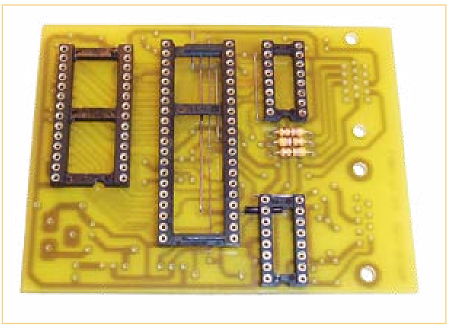


Figure 5. Circuit diagram of the VGA add-on board. Do note the connections of A7, A9 and A11 on IC2.



Fitting the components

Start by fitting the wire jumpers (watch out for shorts with jumper 1 located underneath IC3, it's wise to sleeve it before fitting), then the resistors, diodes, IC sockets (optional, but recommended, especially for the microcontroller IC1), and lastly the capacitors, then the other components.
The DSP reset circuit (R3/D3/C4) is only really required if you power the board from a +5 V supply that takes too long to establish at power on. In most cases, it can simply

Table 1. Two options for the serial link.

VGA add-on	RS-232 (PC)	TTL (µC)
K3 pin 3	TX	-
K3 pin 5	GND	-
K5 pin 1	-	TX
K5 pin 2	-	GND

Table 2. Two options for powering the VGA add-on board.

VGA add-on	+9 V (external)	+5 V (application)
K1 pin 1	+9 V	-
K1 pin 3	GND	-
K5 pin 3	-	+5 V
K5 pin 2	-	GND

be replaced by a jumper fitted in place of R3. The components R11/R12/D4/T1/K3 are used solely for the RS-232 link. Capacitor C11 can be omitted, it makes it possible to slightly offset the line synchronization signal (and hence the left-hand edge of the screen) should this prove necessary.

Resistors R8, R9, and R10 set the levels of red, blue, and green respectively to maximum using a value of 470 Ω. If you fit all three resistors, your image will be in white. Depending on the value of these three resistors (between 470 Ω and a few thousand ohms), you can make your display any color you like. If you fit only R9, you'll get green in true 80s monitor style! Do watch out, though, as some of the latest LCD monitors measure the impedances on the RGB pins of K4, and so won't tolerate the absence of a resistance. To avoid this problem, all you need do is return any of the RGB terminals that are unused to ground using a 470 Ω resistor.

Capacitors C1, C2, and C6 are fitted beneath the dsPIC (IC1, **Figure 6**) socket — another reason why this socket is strongly recommended! Without a socket, you'll have to solder these three capacitors on the track side of the board.
The 15-pin sub-D connector may be salvaged from an old PC video card. You can also do without this connector, but the connections between the video cable and the PCB will need to be made extremely carefully, particularly in terms of the various cable screens.
The exact type of the RAM is not very important, you should choose a 32 KB type with low power consumption and an access time less than or equal to 100 ns. We've even managed to make the board work with a 120 ns RAM. Good results were obtained using a Toshiba TC55257BPL-10, a Hyundai HY62C256LP-10, and a Fujitsu MB84256-12L. Using this type of RAM, the board consumes around 50 mA.
The type 166 FIFOs can be either TTL LS or CMOS 74HC166. For the dsPIC, the dsPIC30F3011-30 version is strongly recommended.

Tables 1 and 2 summarize the connections to the board for the serial port and power supply.

Testing the board

When power is first applied, the RS-232 / VGA converter is initialized in terminal mode, 9,600 bauds, 'scroll' mode, cursor position initialized at the bottom left of the screen. The text resolution is 60 lines by 64 columns, with 8 × 8 pixel characters. In this mode, it can only receive text and the control codes CR and LF, along with the 'fixed mode' control code. The character set includes the standard ASCII codes between <space> (0x20) and the tilde character (~, 0x7E) and a semi-graphics set between 0x80 and 0xFF.

Several types of commands are accepted, like standard terminal-type control codes (RC, BS, VT, LF, etc.), displayable characters, and Escape (ESC) type commands that let you use the VGA mode.
Upon receiving the 'fixed mode' code, the board goes into text and graphics mode. It is then able to receive text and one-byte control codes between 0x01 and 0x20, and high-level commands making it possible to plot graphics objects. The text and graphics can be completely mixed.
A demonstration mode lets you see the board working without sending any commands. Connect the board to a monitor via K4, and set it into demo mode (a jumper across pins 1 and 2 on K5). Connect a +9 V PSU to K1, the LED should light and the demo should start.

You can center the image on your screen by using the horizontal position adjustment. Better still, use the clock adjustment, if your monitor has one, which will make it possible for the image to fill the whole screen.
For testing the board with a computer, connect the board to a monitor via K4, if necessary remove the jumper on K5, connect a PC with RS-232 via K3, and connect a +9 V PSU to K1. Run an ASCII terminal on the PC (we recommend the excellent RealTerm) configured to 9,600 baud, no parity, 8 data bits, one stop bit, no flow control. Type some characters on the PC keyboard, they should appear on the bottom line of the screen. You don't have a COM port on your PC? No problem, any off-the-shelf USB / RS-232 adaptor cable will do the job.
If both these tests are OK, your graphics board is working. To take advantage of all its functions, it's essential to read the user manual [1] which details the command format. You'll find in it some examples showing you how to plot graphics objects in a few seconds, and how to go about creating and previewing your graphics applications using a simple text file, without needing to write a single line of microcontroller code.

Command format and control codes

The board accepts two command formats (ASCII and binary) which are made up of several bytes. The ASCII format is the most suitable for sending commands using a terminal on a PC, it lets you 'get the hang of things' easily and display the result of a command without writing any code.
The binary format is more suitable for commands coming from a microcontroller. Nevertheless, both formats can be used, whether directly from the keyboard or via the intermediary of your application, the choice is yours.

The control codes consist of a single character whose value is located between 0x01 and 0x1B; hence some are difficult to send directly from your PC's keyboard. However, they can be saved in text files and sent via the terminal. These make it possible to manage the display position (cursor), to switch between scroll or fixed modes, and overall screen management (ON, OFF, delete).

The high-level commands are accessed via an ESC type sequence (0x1B). The high-level commands use co-ordinates of text, graphics, or mixed type (see box). All the commands start with the ESC character (0x1B), followed by a letter that determines the command to be used, then a number of characters that depends on the type of command. They make it possible to manipulate text and graphics objects:

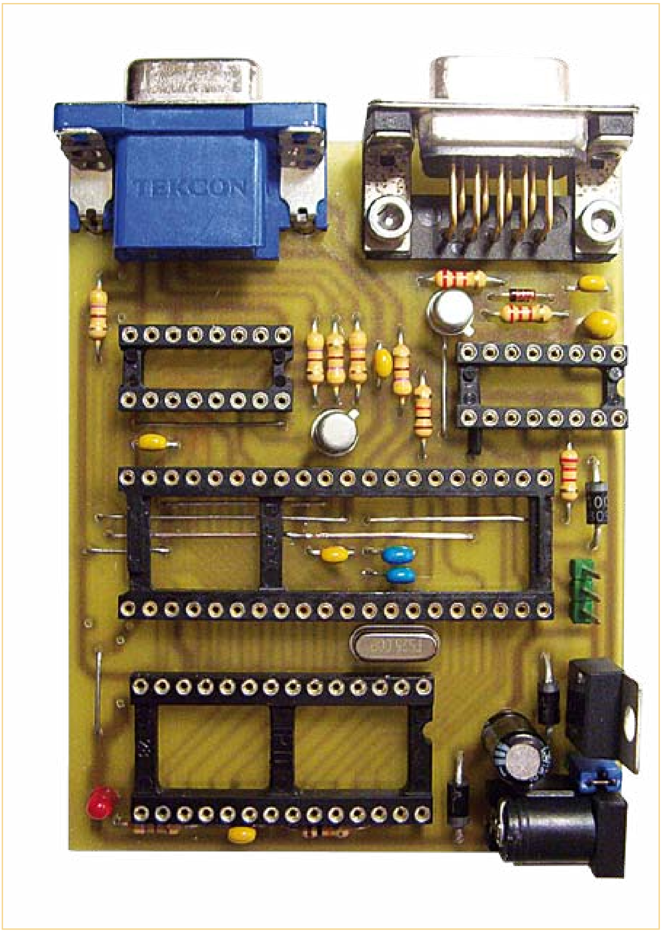


Figure 6. One component may be hiding behind another. The use of a socket for IC1 is strongly recommended.

- set the cursor position;
- draw a circle;
- fill an area with a pattern;
- draw a line;
- turn a pixel on/off;
- draw a rectangle;
- write a string of characters.

Refer to the user manual [1] for details of the control codes and commands.

These commands and control codes can be saved in text files and sent to the board in sequence, making it possible to construct, test, and refine a large part of a graphics project without needing to compile the least line of code on a microcontroller.

Technical support

There's a great deal of information and annotated examples in the technical manual. The author is at your disposal for any questions concerning this project (PCB, programming the dsPIC, project developments) by e-mail at: etiennes33@yahoo.fr

Conclusion

The field of application for this project is huge, from a simple monitor for debugging to simultaneous display of graph plots. This VGA board will never replace a graphical LCD display in on-board systems, but it will allow designers to have a convenient display area at a ridiculously cheap price.

(100147)

Internet Link

[1] www.elektor.com/100147

Advertisement

WIRELESS MADE SIMPLE®

QUICKLY AND COST-EFFECTIVELY MAKE YOUR PRODUCT WIRELESS

RF MODULES
LOW COST • LONG RANGE
TRANSMITTERS • RECEIVERS • TRANSCEIVERS

INTEGRATED CIRCUITS
ENCODERS • DECODERS • TRANSCODERS
COMMAND • CONTROL • KEYLESS ENTRY

OEM PRODUCTS
EASILY CUSTOMIZED

ANTENNAS
FROM WHIPS TO CHIPS

Featured Products:

LT Series Transceiver
- 315 MHz
- 418 MHz
- 433 MHz

- Data Rates to 10K bps
- Direct Serial Interface
- Low Power Consumption
- Low Cost
- Long Range
- Ultra Compact

Ideal for the bi-directional wireless transfer of serial data, control, or command information in the favorable 260-470MHz band. Its advanced synthesized architecture delivers outstanding stability and frequency accuracy.

Function Relay Module
Provides four latched or momentary relay outputs that are capable of switching external AC or DC powered loads of up to 5 amps.
- 315 MHz
- 418 MHz
- 433 MHz
- Dual Power Inputs
- 256 Unique Addresses
- Wide Operational Voltage
- FCC and Industry Canada Pre-Tested

LINX TECHNOLOGIES
WIRELESS MADE SIMPLE
800-736-6677
159 Ort Lane
Merlin, OR 97532
linxtechnologies.com

AVR and 8051 drive 2-color LEDs

By Vladimir Mitrovic (Croatia)

Two-color LEDs consist of two LED chips, usually red and green, encapsulated in one 2- or 3-wire case. Inside the 2-wire case two LED chips are connected in anti-parallel, so the LED color gets determined by the current direction. In the 3-wire case two LEDs share a common wire for their cathodes, while the anodes are connected separately. The latter allows setting different currents through the diodes and producing not only a pure red or green light but also various color blends. The same effect can be obtained with a 2-lead LED as well if you power it from an alternating current source and provide separate current regulation for both current directions. Microcontrollers can efficiently drive 2-color LEDs producing two complementary PWM signals.

AVR example

As shown in **Figures 1 and 2**, an ATtiny13 AVR microcontroller can make a two-color LED produce several different colors using just one current-limiting resistor and no software overhead. Two complementary PWM signals are present at the OC0A and OC0B outputs. When OC0A is at logic 0, OC0B will be at logic 1 and vice versa. The positive and negative time interval ratio at each output determine the average current through the corresponding diode, hence the intensity of light produced. As the signals are complementary, turning up one LED will quench the other. The overall intensity stays approximately the same, but the color changes.

PWM signals are produced by 8-bit Timer/Counter0 with two output compare registers, OCR0A and OCR0B. Timer/Counter0 is configured to run in the Phase Correct PWM Mode. In this mode, the counter counts repeatedly from bottom (0) to top (255) and then from top to bottom. The OC0A output is assigned to the non-inverting, and OC0B to the inverting compare output mode. Therefore, OC0A is cleared on the compare match between the counter and OCR0A while counting up, and set on the compare match while counting down, whereas the OC0B output is inverted. If the values in the OCR0A and OCR0B registers are the same, a complementary PWM signal is produced at the OC0A and OC0B outputs.

In BascomAVR, Timer/Counter0 is configured for PWM by:

```
Config Timer0 = Pwm , Prescale
= 8 , Compare A Pwm = Clear Up
, Compare B Pwm = Clear Down
```

Unfortunately, a bug was found in BascomAVR 1.11.9.1 Demo, causing the command to produce an opposite effect from what's expected: OC0A is actually set while counting up, and cleared while counting down, and the same inversion applies to OC0B. The effect of this bug is that the LED behaves as if it's inversely connected, i.e. with red and green swapped. The proper assembler code would be:

```
ldi r24, $02
!out tccr0b, r24
ldi r24, $B1
!out tccr0a, r24
```

Both examples not only configure the PWM generator but also start it with a frequency of about 2 kHz. Pulse duration and LED color are defined by the value set in the OCR0A and OCR0B registers (remember, the same value must be written in both registers!):

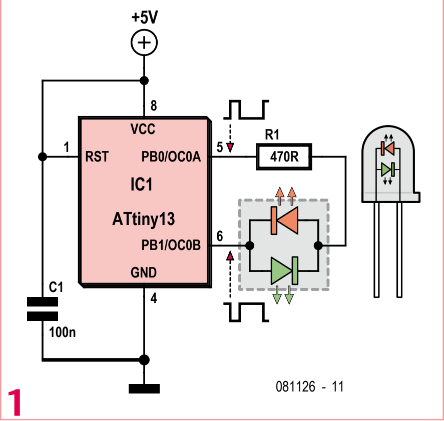
- '0' will stop the PWM generation with OC0A set to '0' and OC0B set to '1' — the green LED will be on and the red one will be off;
- '255' will stop the PWM generation with OC0A set to '1' and OC0B set to '0' — the red LED will be on and the green one will be off;
- any value between 1 and 254 starts the PWM generator — a higher value gives a longer positive pulse duration at the OC0A output, turning up the red LED.

For example, these parameters

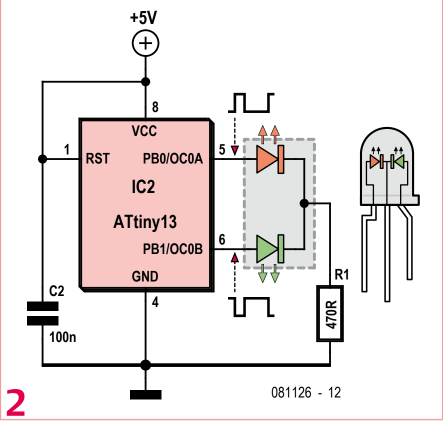
```
OCR0A = 128
OCR0B = 128
```

will cause the same average current through the diodes, producing an orange-colored light. The number of perceivable colors (in practice) and their quality (i.e. how successfully the red and the green components are mixed) depend on several factors: LED opacity, viewing angle, ambient light etc.

A simple BascomAVR program "EE_RGLED_AVR.bas" is given as an example and may be downloaded [1]. The program constantly varies the LED color between red and green through various mixtures of the two colors. The same program works with both LED types. It will work well not only with ATtiny13 but with any other AVR microcontroller that has Timer/Counter0 with



1



2

two output compare registers. When the color is set the program can execute any code while the LED is driven in the background by Timer/Counter0.

8051 example

If you want to drive a two-color LED with an AT89S2051 or any other microcontroller from the 8051 family, you have to consider the architectural differences:

- 8051 microcontrollers have weak pull-ups and therefore cannot source current to a load,
- most 8051 microcontrollers don't have a timer for generating a PWM signal and even the ones that do cannot produce complementary PWM signals without a bit of software.

The first problem is solved with convenient drivers. The simplest solution is shown in **Figures 3 and 4**, where LEDs are driven by the current through resistors R2 and R3. In Figure 3, the microcontroller turns the LED on when the corresponding pin is at logic 0. In Figure 4, the microcontroller turns the LEDs off by short-circuiting them when the corresponding pins are at logic 0. The drawback to either solution is that some current flows through the resistor and I/O port even when the corresponding LED is switched off. As two LEDs are turned on and off alternatively, some 10 mA is constantly wasted. If this is considered a problem, an alternate solution with a pair of buffer gates (for example, type HC125) is shown in **Figure 5**. Two complementary PWM signals are realized in a timer interrupt routine. Timer0 is configured as a 13-bit up counter by:

```
Config Timer0 = Timer , Mode = 0 , Gate =
Internal
On Timer0 Tim0_sub Nosave
Enable Timer0
Enable Interrupts
```

Interrupt routine Tim0_sub is called each time Timer0 overflows. The rate of this event depends on the clock frequency and the initial Timer0 register value. Here, the crystal frequency is 12 MHz and the microcontroller is set to x2 mode, so the clock frequency equals 2 MHz. If the counting starts from 0, it will take 2^{13} counts for the timer to overflow, and the interrupt routine will be initiated approximately once in every 4 ms (244 Hz). If counting starts from a higher number (defined by software), the interrupt routine will be initiated after a shorter time interval.

Byte variable 'Colour' defines the starting value for Timer0. The programmer has to set any value between 0 and 255 and call the 'Set_colour' subroutine, for example:

```
Colour = 64
Gosub Set_colour
```

Depending on the value of 'Colour', 'Set_colour' responds as follows:

eweekly elektor newsletter
30 July 2010
www.elektor.com

Birdhouse made from video tape cassettes
New from Elektor: DVD Audio Collection Vol. 3
Solar energy from window panes
Nonvolatile voice/audio IC with integrated phone functions
10 times better cooling
Energy harvesting development kit available

Magazine
July/August 2010
Subscribe
Forum
In this issue

FLOWCODE 4 Rapid development of prototypes and complex systems

Birdhouse made from video tape cassettes

If from time to time you can't get any fun from electronics at the design or repair level, simply rip old gear apart and use the bits for applications that are impossible or at least difficult to relate to the flow of electrons, the shifting around of logic 0's and 1, or Ohm's Law. In Brussels, birdhouses were spotted made from old VHS video...

Read more

Spotlight
'Green' books

New from Elektor: DVD Audio Collection Vol. 3

We really hate to blow our own trumpet but Elektor is famous for its publications on DIY high end audio. Over the past 30 years or so, many superbly designed amplifiers, preamplifiers, loudspeakers and ancillary [illegible]

- if Colour = 0: stop Timer0, reset P3.0 (red LED off) and set P3.1 (green LED on);
- if Colour = 255: stop Timer0, set P3.0 (red LED on) and reset P3.1 (green LED off);
- if Colour = 1-254: transfer the value in 'Colour' to the upper Timer0 counting register (TH0), reset the lower Timer0 counting register (TL0), reset the color bit 'Colour_bit' and start Timer 0.

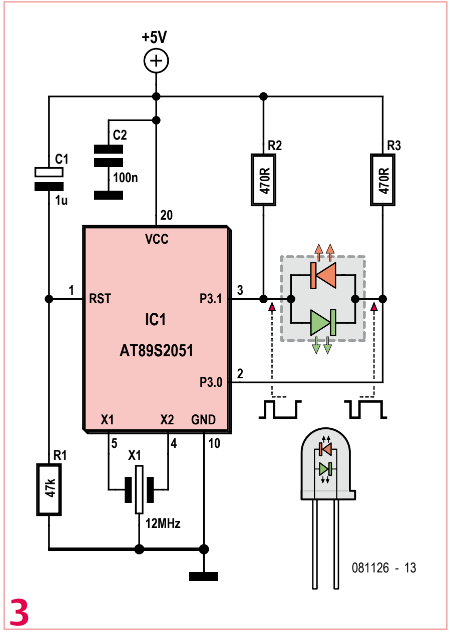
Next, everything happens in the interrupt routine and the program can do whatever is necessary. On every Timer0 overflow the interrupt routine will:

- toggle I/O pins P3.0 and P3.1 that control red and green LED (the program code uses the 'Colour_bit' bit variable to ensure that P3.0 and P3.1 are always complementary);
- complement the 'Colour' variable and transfer the new value to TH0.

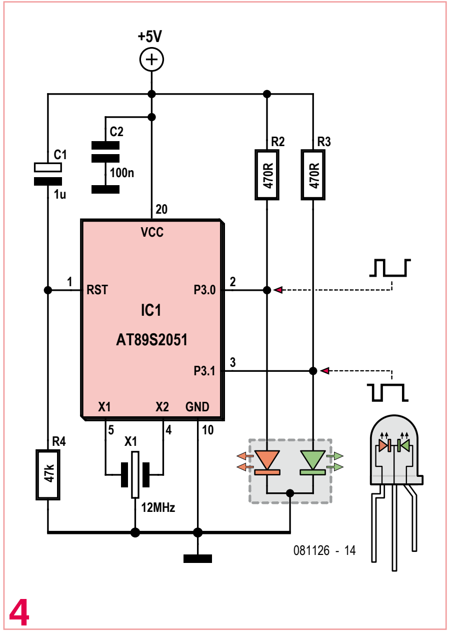
As a result, the variable 'Colour' will have two values alternately: [initialy_set_value] and [255 – initialy_set_value]. According to the previous example, these values would be 64 and 255–64 = 191, causing the red LED to be on for approximately 1/4 and the green LED for 3/4 of the period (i.e. 1 ms and 3 ms). A higher initial value in 'Colour' ensures a stronger red and a weaker green light and vice versa.

A simple Bascom8051 program called 'EE_RGLED_8051.bas' is again given as an example. To have better control over the registers used, the interrupt routine is written in assembler and lasts only 21 cycles. The demo program constantly varies the LED color between red and green through various mixtures of the two basic colors.

The same program works well with both types of 2-color LED. It will also work well with a generic 8051 microcontroller. For microcontrollers that do not support x2 mode (for example, AT89C2051 and the other older microcontrollers), the Set Clkreg.0 command should be deleted. However, this will lower the PWM frequency to about 122 Hz and may cause a slight flickering with some colors. If that's undesirable, use a 20 MHz or 24 MHz quartz crystal.



3



4

The example programs for a 8051 microcontroller might seem a bit awkward compared to the ones for AVRs, but they have certain benefits as well. The most important one is the possibility to freely configure I/O pins that will drive LEDs; just change two 'alias' commands at the beginning of the program:

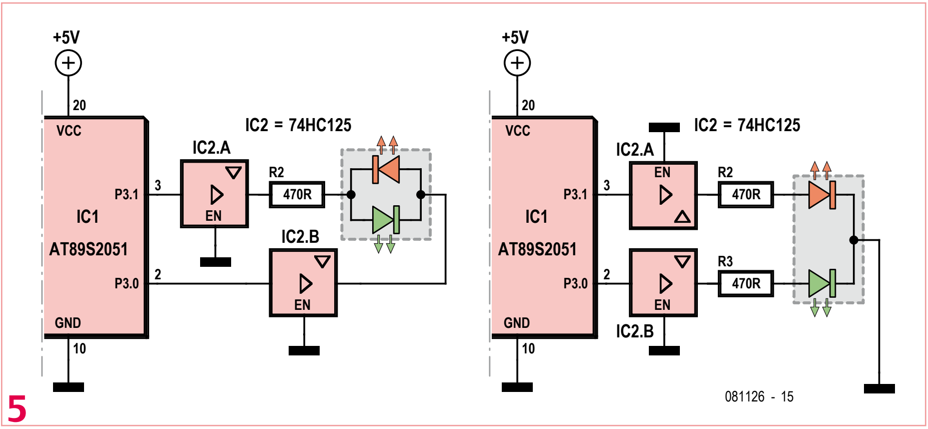
```
Anoda_rd Alias P3.0
Anoda_gn Alias P3.1
```

You may use the second timer, Timer1, to independently drive another RG LED in a similar way.

(081126)

Internet Link

[1] www.elektor.com/081126



5

Stellaris ARM Controller goes Biological

Design considerations for an egg incubator

By Tianyu Chen (People's Republic of China)

This microcontroller circuit was developed in response to challenges posed by a biological process. Although the final application, a reliable system to hatch chicken eggs, may seem specialized, the method of getting there has potential for many other systems where parameters like temperature, motion and humidity have to be monitored and controlled "with hen's diligence".

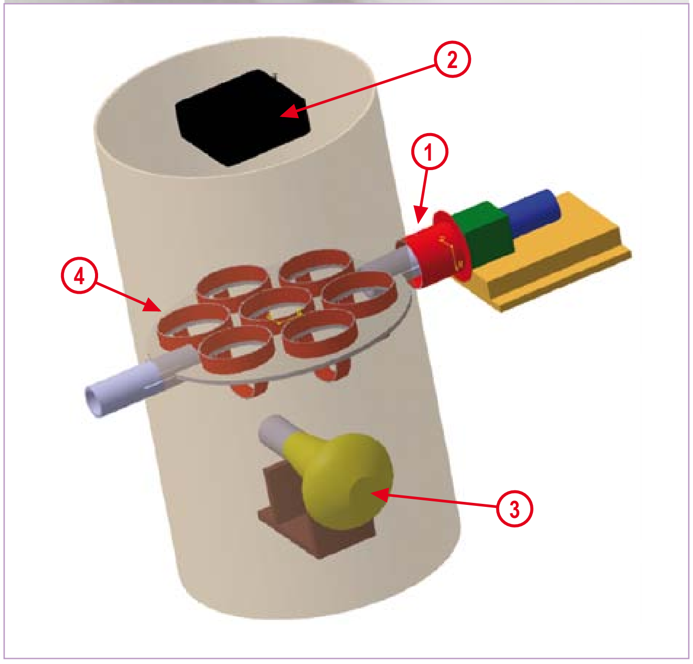


Figure 1. Artist's impression of the egg hatching chamber. 1: motor and coaxial gear drive. 2: PC fan. 3: 25 watts bulb. 4: egg tray.

Mimicking a domestic hen hatching her eggs, i.e. developing an incubator (egg hatching machine) is far from trivial. The task requires mechanical as well as electronics skills in combination with some research into biology. Rather than presenting a ready-engineered product, the aim of this article is to show the process of various factors like real world constraints, electronics, software and mechanics being analyzed and combined.

The real-world factors

You might think that temperature is the most important factor during the process of egg incubation. However, as it turns out, humidity and other factors are also significant. Based on research and literature studies the author was able to list five major points that must be observed, in this order of importance: 1. temperature; 2. relative humidity; 3. rotation; 4. ventilation; 5. sanitation.

Temperature. Because the embryo has no abilities for temperature self-regulation, for it to develop normally an external temperature control device is required. Although some eggs will successfully hatch at a temperature between 35 and 40.5 degrees centigrade, the optimum temperature for the embryos is 37.8 °C.

Although a higher temperature will accelerate embryonic development, it will also cause the death rate to increase and the quality of the chicken to drop. Consequently, the temperature must be as constant as possible around 37.8 °C.

Relative humidity. Normally, the relative humidity during the embryo development will be in the range 40–70%, but the optimum values are between 50% and 60%. If the proper humidity is maintained the egg will absorb heat evenly in the early days of the incubation, and will dissipate heat easily near the end of the incubation period.

Rotation. The eggs should be rotated to prevent embryo adhesion and to stimulate the amnion motion. Ideally the egg should

Note. Readers' Projects are reproduced based on information supplied by the author(s) only.
The use of Elektor style schematics and other illustrations in this article does not imply the project having passed Elektor Labs for replication to verify claimed operation.

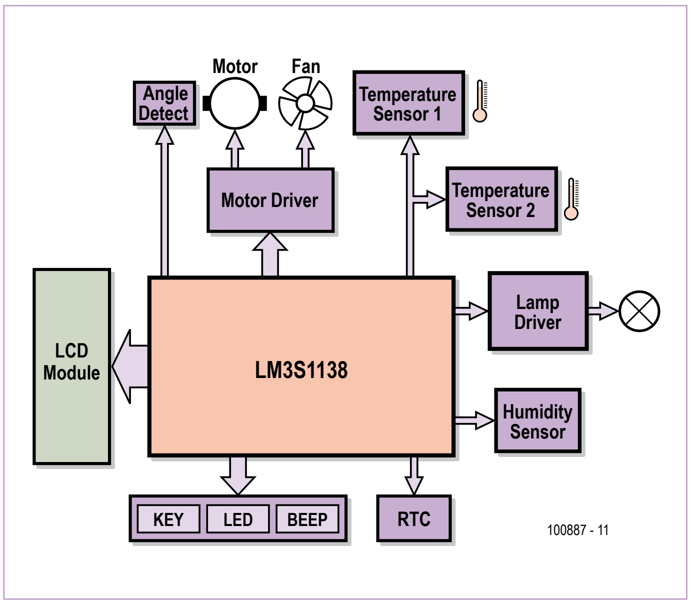


Figure 2. The elements that go into making the control circuit based on an LM3S1138 ARM Cortex-M3 microcontroller.

be turned 6–8 times a day, especially during the first two weeks.

Ventilation. During the embryonic development in the egg chamber, air exchange is always in progress except for a few days at the very beginning. As the embryo grows, the need for oxygen increases. Ideally the oxygen content should be 21%, while carbon dioxide levels should remain below 0.5%. CO_2 values in excess of 1% will cause many embryos to die.

Sanitation. Although the embryo is surrounded by a colloidal substance and protected by a shell and a cuticle, some germs can still get to it, reducing the chances of successful hatching. Consequently a good deal of attention must be paid to sanitation and disinfection of the hatchery system.

A DIY hatching chamber

A lot of effort went into finding a device that could be modified into an egg hatching chamber. The chamber must have good thermal insulation properties, fair ventilation and sufficient space for seven eggs. An old Sanyo electric thermos pot found in the kitchen proved to be the ticket to success.

To be able to link the hardware functionally to the factors discussed above it is shown as an artist's impression in **Figure 1**. On the top, there is a fan (2) salvaged from an old PC. At the center is a tray (4) which can hold the eggs during the hatching period. The tray is driven by a coaxial geared motor (1) in order to turn the eggs at intervals.

The air conditions in the chamber are all important — enough oxygen is needed but the air temperature must be stable. If too much fresh air gets into the chamber at a fast rate, the air temperature is in danger of dropping too fast. A thermal balancing device was found in the form of a standard 25 watts bulb (3) acting as a heat source. As soon as the hot air gets to the top of the chamber, the fan will blow it down and at the same time also draw in a little fresh air from the hole in the cover. With fresh air and hot air mixed to a good degree, the air inside the chamber will have a constant temperature and a high oxygen ratio.

The electronics

After the biological and mechanical considerations comes the electronics, whose function is to make it all happen. At the heart of the control circuit developed sits an LM3S1138 ARM Cortex™ M3-based microcontroller [1] from Luminary Micro (now owned by Texas Instruments). This CPU belongs is from the "Stellaris®" series.

The block diagram in **Figure 2** shows peripherals around the CPU like Key, LED, and 'Beep' that allow the user to set the hatching characteristic and display its status. When the system is started it asks the user to set the kind of egg and the expected hatching time. During the hatching process the LCD displays the current temperature and humidity in the chamber, and 'days remaining to hatch'. If the temperature or humidity is higher or lower than the warning level, or the system has detected an exceptional condition, the beeper sounds, alerting you to take action to prevent embryo death.

The system has two temperature sensors and one humidity sensor designed in. The two temperature sensors are located in such a way as to enable poor ventilation to be detected and eliminated by fan action. Also, if one temperature sensor is damaged or malfunctioning the other acts as a backup.

The actual circuit is spread over two circuit boards. The core board contains the CPU, the debug interface, the reset circuit, key and LED. The other board has peripherals like the RTC and the motor driver. Due to space constraints the full circuit diagram of the system is not included in this article — it is however available as a free download from the Elektor website [2]. For the main components we have! CPU = LM3S1138; motor driver = L298N; RTC = DS1305; humidity sensor = HS1101/NE555; temperature sensors = LM75. In all fairness, nothing too esoteric apart from the Luminary CPU.

The software

A flow chart of the control software is shown in **Figure 3**. All aspects of the mechanical construction, the electronics and the real-world constraints of egg hatching are duly taken into account by a fair bit of coding for a reasonably powerful ARM controller. The source code file for the LM3S1138 can be downloaded free of charge from [2].

(100887)

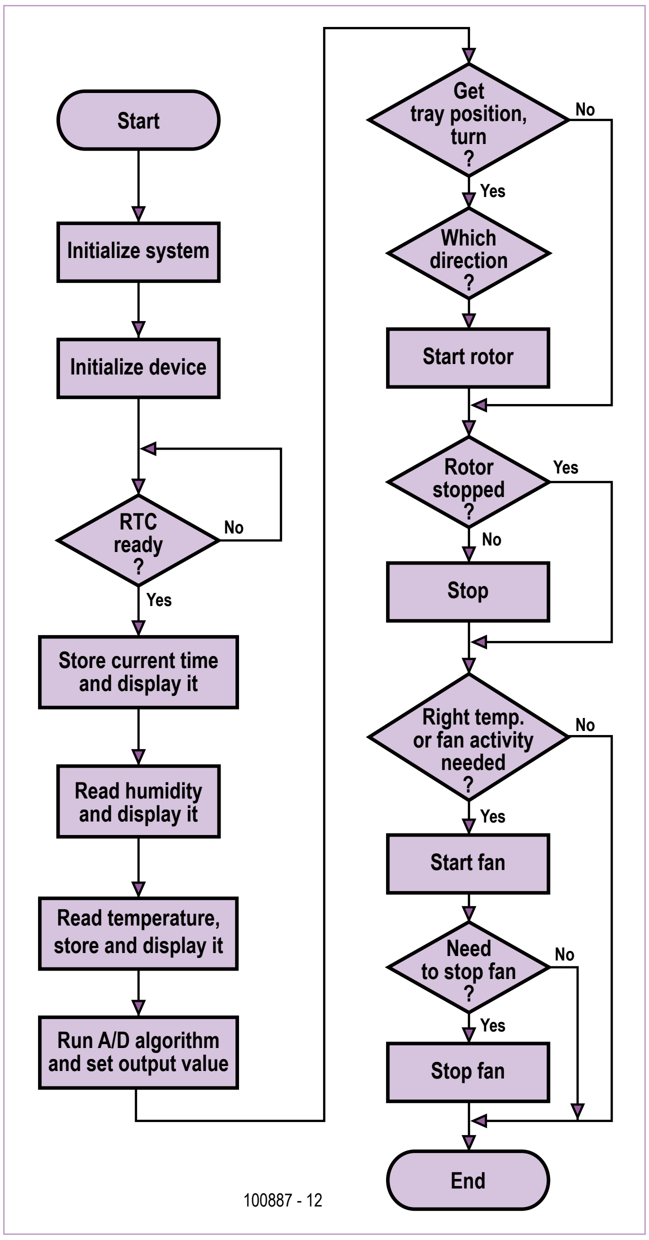

Start
Initialize system
Initialize device
RTC ready ?
No
Yes
Store current time and display it
Read humidity and display it
Read temperature, store and display it
Run A/D algorithm and set output value
Get tray position, turn ?
No
Yes
Which direction ?
Start rotor
Rotor stopped ?
Yes
No
Stop
Right temp. or fan activity needed ?
No
Yes
Start fan
Need to stop fan ?
No
Yes
Stop fan
End
100887 - 12

Figure 3. Flowchart of the program written for the TI (Luminary Micro) "Stellaris" LM3S1138 microcontroller governing the operation of the egg hatching system.

Internet Links

[1] www.luminarymicro.com/products/LM3S1138.html (LM3S1138 datasheet)

[2] www.elektor.com/100887 (circuit diagrams and project software)

Advertisement

PAiA

Back-to-school discounts on basic kits at paia.com

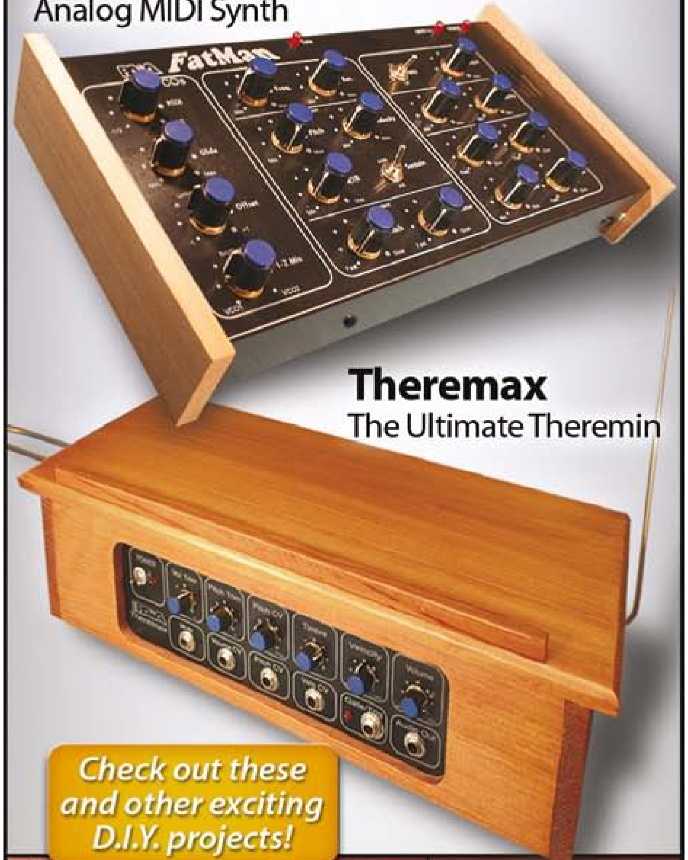
Electronic Kits for Enthusiasts

BUILD YOUR OWN!

PAiA 9700s
Modular Synthesizer

FatMan
Analog MIDI Synth

Theremax
The Ultimate Theremin



Check out these and other exciting D.I.Y. projects!

PAiA paia.com

Music/Tutorials/Kits
Guitar Effects
Tube Electronics
Studio Gear

Those Lying Instruments

Gerard Fonte (USA)

Much of engineering depends upon instrumentation. We rely on the readings we see. But sometimes what we see isn't what we have. It's important to know when not to trust our measurements.

Input Resistance

Take two 10 megohm resistors and connect them in series. Attach the ends of the series to a 9-volt battery and measure the voltage across either resistor. The voltage should be 4.5 volts. It seems simple, except that with nearly all analog meters and most digital meters the reading will be much less.

The reason for this is basic if you realize that your meter is not perfect. There is a finite resistance between the probes. With analog meters this resistance is usually measured in 100's of kilohms. Digital meters typically have 1 to 10 megohms input resistance. This means that when you measure the voltage, you are placing a resistor in parallel with the 10 megohm resistor. This reduces the effective resistance of that resistor and changes the voltage across it. Every piece of test equipment has some input resistance. Know what it is and anticipate odd readings in high resistance circuits.

The good news is that you can usually determine what the actual voltage is. You need to know the resistance of your circuit and the input resistance of your meter. Your meter resistance is found in your owner's manual. The circuit's resistance can usually be estimated by examining the schematic. The formula for two resistors in parallel is:

$$R_p = (R1 \times R2)/(R1+R2).$$

It is also possible for your analog ohmmeter to actually destroy semiconductors or other sensitive parts. This is because it applies a current across the leads (from the internal battery). This current varies according to the meter's resistance setting as well as the 'Ohms per Volt' rating. I measured my meter (20 kohms/volt) on the R-1X scale and found 40 mA passing through the leads. Obviously, 40 mA can make bad things happen (Digital meters generally use a different front-end and don't have this problem). Usually the resistance current is specified in the manual. But it's useful to measure it for yourself. You remember it better if you do it, rather than read it.

Grounding

Proper grounding can make a world of difference. Nowadays, AC powered instruments always have their chassis grounded. So there is a true ground. Some self-powered instruments have a 'ground' connection that is more accurately a 'return' point. This is because this 'ground' is floating and not connected to a real ground point.

In theory, a floating ground allows differential measurements. That is, you can place the two probes anywhere to get voltage (or resistance, etc) between these two points. A plain multi-meter has a floating ground and makes always differential measurements.

An oscilloscope has a ground lead and a probe lead for measurements. This is a single-ended measurement. It is always referenced to a real ground which is almost always the chassis. It's clear that if you connect the oscilloscope ground lead to someplace other than ground, you can get serious problems.

The 'almost always' mentioned above refers to off-line switching power supplies, which are now common in TV's and some high-powered, home-theater amplifiers. They use a 'common' or 'return' which is not a ground. If you connect your oscilloscope ground lead to the chassis and make measurements of the power supply, you will get nonsense. If you connect the ground lead to the 'common' point you will instantly get a huge spark as a kilowatt or so of AC power flows through the ground lead of the oscilloscope. This is usually transitory, depending on how fast the AC circuit breaker acts. But it can be devastating to the TV and/or the oscilloscope.

More mundane problems with grounding lead to distortion and noise pick-up. Noise is the concern with very small signals. Trying to examine a 5 millivolt signal with 50 millivolts of noise (not uncommon with a bad ground) is not trivial. Distortion is the problem with high-frequency signals. You will often see ringing on fast edges. This can cause mis-triggering of the oscilloscope and makes it hard to measure the true timing relationships of different signals.

Frequency Response

The last common problem we'll look at is the frequency response of the test instrument. Digital multi-meters typically top out at 100,000 Hz or so. Curiously, the analog multi-meters can often operate up to 10 MHz, or more. This is because the analog meters are basically resistive voltage dividers that drive a coil of wire. Whereas, the digital meters have an internal amplifier to drive the analog to digital converter. This is a case where simpler is faster.

Don't expect to see much difference between a 25 MHz sine wave versus a 25 MHz square wave with a 50 MHz oscilloscope. The wave is square because of all the higher frequency harmonics. Since the oscilloscope can't display these harmonics, it looks like a sine wave, too. Again, there will be problems with measuring timing relationships.

And don't forget about the 'scope probes! They have to be rated for the frequency, too. I've seen too many instances of 50 MHz probes connected to 300 MHz oscilloscopes and the user wondering why things are strange. If you are fortunate enough to have a good 'scope, spend the extra money for a proper set of probes. Otherwise, you really only have a low-frequency 'scope.

Conclusion

There are many other factors that can cause your equipment to lie to you such as wave-shape or temperature. It's important to always remember that there is no such thing as a perfect measurement. In fact, some measurements are technically impossible. For example, in order to measure power properly, you need to measure the voltage and current simultaneously. But measuring the voltage requires some small amount of current into the meter. And to measure the current, you have to insert a small series resistor which reduces the voltage. These errors may be insignificant and it may be possible to calculate what they are. Nevertheless, the errors are there.

(110430)

Hexadoku

Puzzle with an electronics touch

Here at Elektor Castle a good deal of spring sunshine was bestowed on us and we hope you too can enjoy quiet, warm afternoons sitting on a terrace with coffee or a beer within arm's reach and this here Hexadoku puzzle in sight. Get cracking and enter the right numbers in the puzzle. Next, send the ones in the grey boxes to us and you automatically enter the prize draw for one of four Elektor Shop vouchers. Have fun!

The instructions for this puzzle are straightforward. Fully geared to electronics fans and programmers, the Hexadoku puzzle employs the hexadecimal range 0 through F. In the diagram composed of 16 × 16 boxes, enter numbers such that **all** hexadecimal numbers 0 through F (that's 0-9 and A-F) occur once only in each row, once in each column and in each 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. Correct entries received enter a draw for a main prize and three lesser prizes. All you need to do is send us the numbers in the grey boxes.

Solve Hexadoku and win!

Correct solutions received from the entire Elektor readership automatically enter a prize draw for one Elektor Shop voucher worth $ 140.00* and three Elektor Shop Vouchers worth $ 70.00* each, which should encourage all Elektor readers to participate.

*Subject to exchange rate.

Participate!

Before July 1, 2011, send your solution (the numbers in the grey boxes) by email, fax or post to

Elektor Hexadoku – 4 Park Street – Vernon CT 06066

USA.

Fax 860 8751-0411

Email: hexadoku@elektor.com

Prize winners

The solution of the April 2011 Hexadoku is: B9A65.

The Elektor $140.00 voucher has been awarded to M. Borias (The Netherlands).

The Elektor $70.00 vouchers have been awarded to Bertrand Danet (France), Sven Skjenneberg (Norway), Ralf Möckel (Germany).

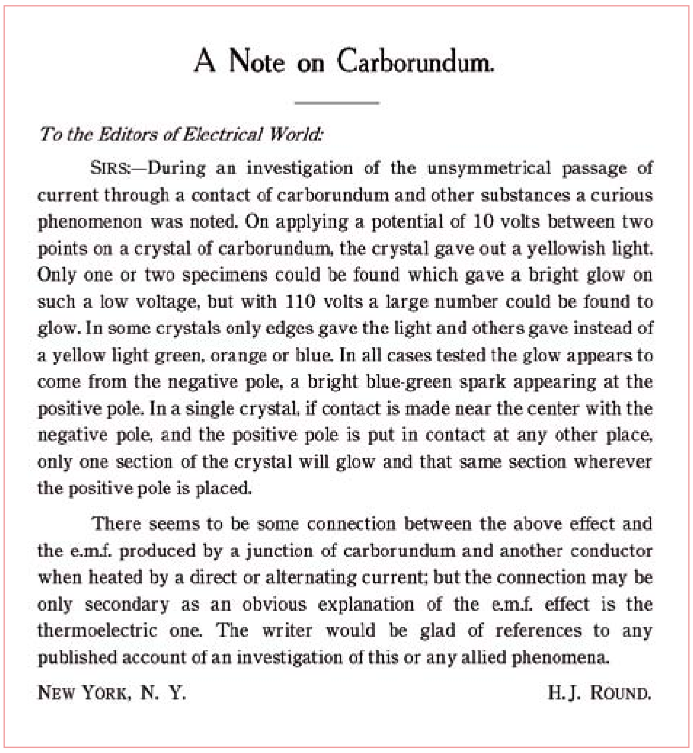
Congratulations everyone!

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

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

Solid Light: the Remarkable

by Andrew Emmerson (UK)


A Note on Carborundum.

To the Editors of Electrical World:

SIRS:—During an investigation of the unsymmetrical passage of current through a contact of carborundum and other substances a curious phenomenon was noted. On applying a potential of 10 volts between two points on a crystal of carborundum, the crystal gave out a yellowish light. Only one or two specimens could be found which gave a bright glow on such a low voltage, but with 110 volts a large number could be found to glow. In some crystals only edges gave the light and others gave instead of a yellow light green, orange or blue. In all cases tested the glow appears to come from the negative pole, a bright blue-green spark appearing at the positive pole. In a single crystal, if contact is made near the center with the negative pole, and the positive pole is put in contact at any other place, only one section of the crystal will glow and that same section wherever the positive pole is placed.

There seems to be some connection between the above effect and the e.m.f. produced by a junction of carborundum and another conductor when heated by a direct or alternating current; but the connection may be only secondary as an obvious explanation of the e.m.f. effect is the thermoelectric one. The writer would be glad of references to any published account of an investigation of this or any allied phenomena.

NEW YORK, N. Y. H. J. ROUND.

Figure 1. H.J. Round's report in *Electrical World*, 1907.

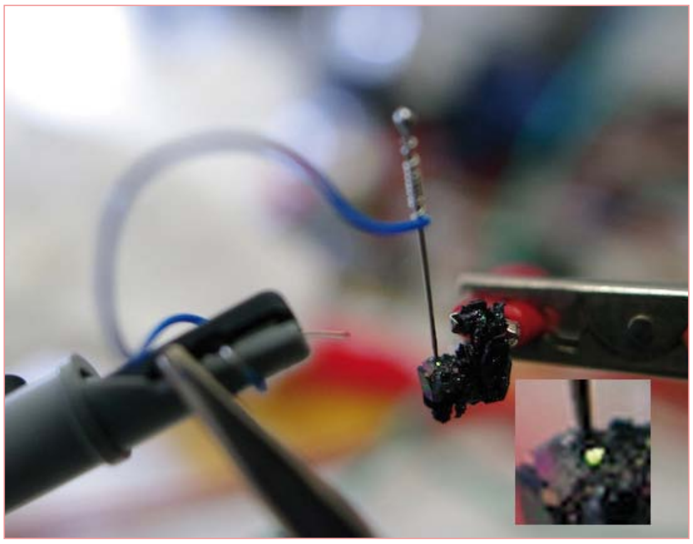


Figure 2. Michael Lippert has replicated Round's experiments of 1907 that revealed the effect underlying the light emitting diode (LED). Here a negatively charged needle contacts a positively charged crystal of silicon carbide or carborundum. At a voltage of 9 V and a current of 30 mA a green glow can be observed at the contact point between needle and pin. (Wikimedia Commons)

Who invented the light emitting diode or LED? Nobody! Let's rephrase the question then; who was the first to discover light emission from semiconductor substances? The answer is Marconi's research assistant, Henry (H. J.) Round, in 1907. But Round was busy with other work and it was left to others to elaborate — or rediscover independently — his findings.

His report (**Figure 1**), compiled while preparing cat's whisker detectors (diodes) for radio experiments, notes that a direct current potential applied between two points on a crystal of carborundum (silicon carbide, SiC) caused the crystal to produce light. He proposed a possible link between the voltage across the carborundum junction and the light emission.

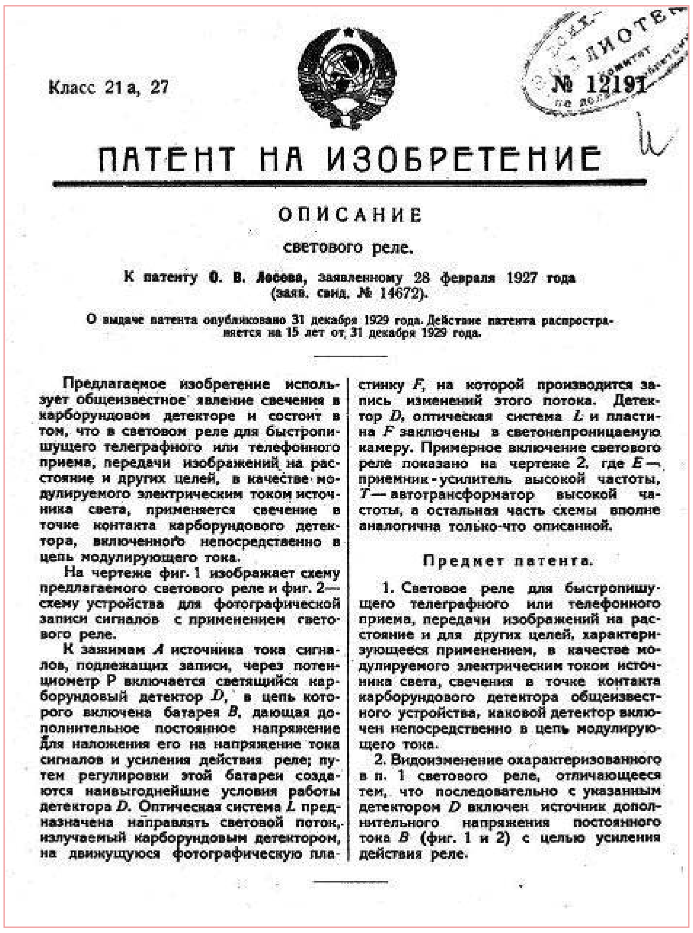
At the time Round's discovery attracted little interest, although 102 years later it inspired German experimenter Michael Lippert to recreate these trials (**Figure 2**). You can see more of these photos and practical details on his blog [1]. Unlike some 'don't try this at home' exploits, Michael's experiment is deceptively simple. All you need is a 9 V power supply, a pin (forming a cat's whisker detector), a carborundum crystal (which Michael bought cheaply on eBay) and a darkened room. Move the pins slowly across the surface of the crystal and you should be rewarded with a tiny glow of colored light at the contact point. Depending on the spot you have chosen and the amount of current flowing, your home-made LED may appear in orange, yellow or green. It can be quite bright, states Michael, even if a bit hit-and-miss.

Records of light emission from semiconductors next appear in the early 1920s, when a self-taught Russian radio technician by the name of Oleg Losev (**Figure 3**), also written as Losov or Lossev and pronounced Olyeg Lossov, made the same discovery as Round before him. In other words, he noticed that crystal diodes used in radio receivers emitted light when current was passed through them. Unlike Round, however, he investigated the subject of electroluminescence in depth at the Physico-Technical Institute in Leningrad, publishing a number of scientific papers discussing to the current-voltage characteristics of SiC diodes and their relation to light emission. One of these, 'Luminous carborundum detector and detection effect and oscillations with crystals', in the British periodical *Philopsophical Magazine* (November 1928) ran to 21 pages.

Whereas Round saw electroluminescence only as a curiosity, Losev had no doubts over the practical significance of his discovery. Miniature non-vacuum light sources that operated at low voltages (less than 10 V) and at very high speed could perform tasks that normal light bulbs could not and his 1927 patent for a 'light relay' foresaw its use "for fast telegraphic and telephone communication, transmission of images and other applications when a light luminescence contact point is used as the light source connected directly to a cir-

Prehistory of the LED

Figure 3. Oleg Losev, the Russian radio technician who noticed that diodes used in radio receivers emitted light when current was passed through them. (Wikimedia Commons)



Класс 21 а, 27

№ 12191

ПАТЕНТ НА ИЗОБРЕТЕНИЕ

ОПИСАНИЕ

светового реле.

К патенту О. В. Лосева, заявленному 28 февраля 1927 года (заяв. свид. № 14672).

О выдаче патента опубликовано 31 декабря 1929 года. Действие патента распространяется на 15 лет от 31 декабря 1929 года.

Предлагаемое изобретение использует общеизвестное явление свечения в карборундовом детекторе и состоит в том, что в световом реле для быстропишущего телеграфного или телефонного приема, передачи изображений на расстояние и других целей, в качестве модулируемого электрическим током источника света, применяется свечение в точке контакта карборундового детектора, включенного непосредственно в цепь модулирующего тока.

На чертеже фиг. 1 изображает схему предлагаемого светового реле и фиг. 2—схему устройства для фотографической записи сигналов с применением светового реле.

К зажимам *A* источника тока сигналов, подлежащих записи, через потенциометр P включается светящийся карборундовый детектор *D*, в цепь которого включена батарея *B*, дающая дополнительное постоянное напряжение для наложения его на напряжение тока сигналов и усиления действия реле; путем регулировки этой батареи создаются наивыгоднейшие условия работы детектора *D*. Оптическая система *L* предназначена направлять световой поток, излучаемый карборундовым детектором, на движущуюся фотографическую пластинку *F*, на которой производится запись изменений этого потока. Детектор *D*, оптическая система *L* и пластина *F* заключены в светонепроницаемую камеру. Примерное включение светового реле показано на чертеже 2, где *E*— приемник-усилитель высокой частоты, *T*— автотрансформатор высокой частоты, а остальная часть схемы вполне аналогична только-что описанной.

Предмет патента.

1. Световое реле для быстропишущего телеграфного или телефонного приема, передачи изображений на расстояние и для других целей, характеризующееся применением, в качестве модулируемого электрическим током источника света, свечения в точке контакта карборундового детектора общеизвестного устройства, каковой детектор включен непосредственно в цепь модулирующего тока.

2. Видоизменение охарактеризованного в п. 1 светового реле, отличающееся тем, что последовательно с указанным детектором *D* включен источник дополнительного напряжения постоянного тока *B* (фиг. 1 и 2) с целью усиления действия реле.

Figure 4. USSR patent issued to Oleg Losev in 1927 for his 'light relay' system (courtesy PatentsFromRU.com).

cuit of modulated current" — in other words the optoelectronic systems that make modern communications possible. Tragically his promising investigations ended in 1942 when he died of starvation in the Siege of Leningrad, aged only 39.

So do Round and Losev deserve more credit for their discoveries with light emitting semiconductors? Probably not, as they led (no pun intended!) to a dead end. As Wikipedia explains, silicon carbide is an inefficient material for light-emitting diodes. Thus neither Round's nor Losev's pioneering work, even if it had been continued, was likely to lead to practical or indeed commercial success.

(110021)

Internet Link

[1] www.dlip.de/?p=99 Michael Lippert builds a DIY LED from SiC.

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.com

ELEKTOR SHOWCASE

To book your showcase space contact Strategic Media Marketing Inc.

Tel. 1-978-281-7708 | **Fax 1-978-281-7706** | **Email ElektorUSA@smmarketing.us**

HAMEG® Instruments
A Rohde & Schwarz Company

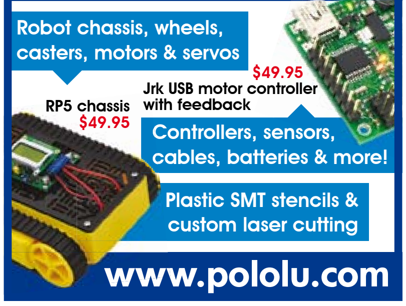
- ☑ Oscilloscopes
- ☑ Power Supplies
- ☑ Spectrum Analyzers
- ☑ RF Instruments
- ☑ Programmable Measuring Instruments

HAPRO Electronics
Tel: +1-516-794-4080
sales@haproelectronics.com
www.hameg.us

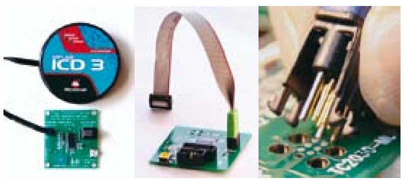
Build Your Own:
THEREMIN
Theremax
The Ultimate Theremin

PAiA paia.com
Makers since 1959
Music/Tutorials/Kits
Analog Synthesizers
Guitar Effects
Tube Electronics
Studio Gear

Pololu Robotics & Electronics
Save 10% on motor controllers with coupon code ELEKTORMC94


Robot chassis, wheels, casters, motors & servos
$49.95
Jrk USB motor controller with feedback
RP5 chassis $49.95
Controllers, sensors, cables, batteries & more!
Plastic SMT stencils & custom laser cutting
www.pololu.com

Tag-Connect


JTAG Cables Plug Directly To PCB!
The GREENEST Connector is NO CONNECTOR!!!
Save Space & Cost on every Board!!
www.PlugOfNails.com

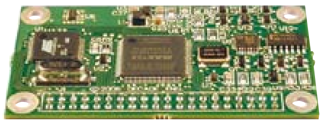
INTEGRATED
PLCs with Integrated Ethernet

From $129

- Serial ports
- MODBUS TCP/IP
- Digital and Analog I/Os
- PWM/PID/Stepper Control

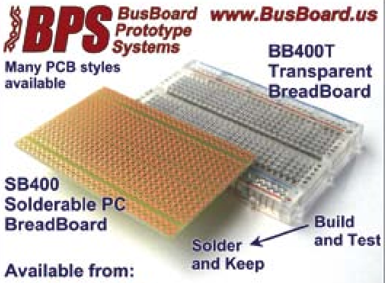
tel : 1 877 TRI-PLCS
web : www.tri-plc.com/ek.htm
TRIANGLE RESEARCH INTERNATIONAL

bob-4
basic overlay board for NTSC & PAL


NEW REDUCED PRICING
Automatic text scroll & crawl · 'TTL-232' & SPI control ports
Any size for custom fonts · Vector & bitmap graphics
DECADE ENGINEERING
www.decadenet.com

Apex Wireless
Your Home for Wireless Design Services
Electrical
Mechanical • Certification
(303) 443-6699
www.apexwireless.com
Boulder, Colorado

BPS BusBoard Prototype Systems www.BusBoard.us
Many PCB styles available


BB400T Transparent BreadBoard
SB400 Solderable PC BreadBoard
Build and Test
Solder and Keep
Available from:
JAMECO ELECTRONICS
amazon.com
MOUSER ELECTRONICS a tti company

SCIENCE, ROBOTICS & ELECTRONICS
IMAGES Scientific Instruments Inc.
Microcontrollers
Servo Motor Controllers
Artificial Vision
Speech Recognition
Flex Sensors
www.imagesco.com
Tele: (718) 966-3694 Fax: (718) 966-3695

AP CIRCUITS
PCB Fabrication Since 1984

As low as...	Two Boards
$9.95 each!	Two Layers
	Two Masks
	One Legend

Unmasked boards ship next day!
www.apcircuits.com
VISA MasterCard PayPal IPC MEMBER BBB

MaxSonar
Ultrasonic Range Finder

XL-MaxSonar-EZ
- Beam pattern choice
- High acoustic power
- Real-time calibration
- $39.95USD qty. 1

MaxSonar-WRC IP67
- Compact packaging
- Standard ¾" fitting
- Quality narrow beam
- $99.95USD qty. 1

www.maxbotix.com

TO BOOK YOUR SHOWCASE SPACE CONTACT STRATEGIC MEDIA MARKETING INC.
Tel. 1-978-281-7708 • Fax 1-978-281-7706
Email ElektorUSA@smmarketing.us

SHOWCASE YOUR COMPANY HERE

Elektor provides you with more ways to promote your business with our Showcase section -- a permanent feature of the magazine where you will be able to further highlight your products and services.

- For just $1100 ($100 per issue for eleven issues), Elektor will publish your company name, website address and a 30-word description.
- For $1650 ($150 per issue for eleven issues), we will publish the above plus run a 2-inch square, full-color image – i.e., product shot, screen shot from your website, company logo, catalog cover – or whatever you choose to showcase.

Places are limited and spaces will go on a strictly first come, first served basis. So please fax back your order today!

I wish to promote my company, please book my space:
• Text insertion only for $1100 • Text and photo for $1650

NAME: .... ORGANIZATION: ....
JOB TITLE: ....
ADDRESS: ....
....
.... TEL: ....
PLEASE COMPLETE COUPON BELOW AND FAX BACK TO 1-978-281-7706
COMPANY NAME ....
WEB ADDRESS ....
30-WORD DESCRIPTION ....
....
....
....
....

Going Strong

A world of electronics from a single shop!

Fundamental Amplifier Techniques with Electron Tubes
Theory and practice with design methods for self construction
elektor
THUMP! 800+ pages
Limited Period Offer for Elektor Subscribers!
16% DISCOUNT
www.elektor.com/june

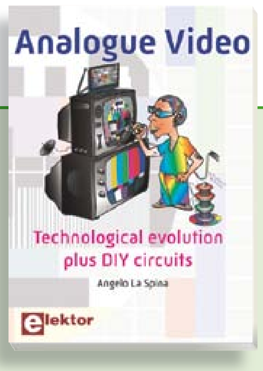
Books

The ultimate tube amplifier reference book!

Fundamental Amplifier Techniques with Electron Tubes

This book is a must-have for all tube fans and the growing circle of RAFs (retro-audio-aficionado's). In a mind blowing 800+ pages Rudolf Moers covers just about everything you need to know about the fundamentals of electron tubes and the way these wonderful devices were designed to function at their best in their best known application: the (now vintage) tube audio amplifier. The aim of the book is to give the reader useful knowledge about electron tube technology in the application of audio amplifiers, including their power supplies, for the design and DIY construction of these electron tube amplifiers. This is much more than just building an electron tube amplifier from a schematic made from the design from someone else: not only academic theory for scientific evidence, but also a theoretical explanation of how the practice works. No modern simulations, but because you first have to understand the circuit calculations, then you can work with your hands to build the circuit and last, but not least, if you have a multimeter, a signal generator and an oscilloscope, you can measure the circuit parameters yourself to see that theory and practice are very close. This very aim and coverage makes the book a unique reference source.

834 pages • ISBN 978-0-905705-93-4 • $104.90



Technological evolution plus DIY circuits

Analogue Video

This book is intended for electronics enthusiasts and professionals alike, who want a much deeper understanding of the incredible technology conquests over the pre-digital decades that created video. It details evolution of analogue video electronics and technology from the first electro-mechanical television, through advancements in Cathode Ray Tubes, transistor circuits and signal processing, up to the latest analogue, colour-rich TV, entertainment devices and calibration equipment.

222 pages • ISBN 978-0-905705-96-5 • $42.80

Solutions for control system applications

Introduction to Control Engineering

This book is intended as a source of reference for hardware and software associated with instrumentation and control engineering. Examples are presented from a range of industries and applications. Throughout the book, circuit diagrams and software listings are described, typical of many measurement and control applications. The hardware and software designs may be used as a basis for application by the reader. The book contains examples of PIC, PLC, PAC and PC programming.

164 pages • ISBN 978-0-905705-99-6 • $44.40

RFID, NFC, Zigbee, GPS and more

DVD Wireless Toolbox

On this DVD-ROM you'll find a number of technical documents and tools that will enable you to add wireless data exchange to your electronics systems. The choice of equipment depends on the transmission distance: a few centimetres using Near Field Communication (NFC) or Radio Frequency Identification (RFID), tens of metres with the Bluetooth, Wi-Fi or ZigBee systems, or indeed thousands of kilometres using a module for receiving GPS data. The DVD contains technical documentation (spec. sheets, application notes, user guides, etc.) on various devices according to the frequency and/or protocol used. All of the documents are PDF files. Finally, this Wireless Toolbox DVD contains a collection of articles on this topic that have appeared in Elektor magazine.

ISBN 978-90-5381-268-6 • $46.00

Bestseller!

All articles in Elektor Volume 2010

DVD Elektor 2010

This DVD-ROM contains all editorial articles published in Volume 2010 of the six language versions of Elektor. Using the supplied Adobe Reader program, articles are presented in the same layout as originally found in the magazine. An extensive search machine is available to locate keywords in any article. With this DVD you can also produce hard copy of PCB layouts at printer resolution, adapt PCB layouts using your favourite graphics program, zoom in / out on selected PCB areas and export circuit diagrams and illustrations to other programs.

ISBN 978-90-5381-267-9 • $37.90

Bestseller!

Pico C Meter

(April 2011)

RF and radio repair fans probably do need to be told, but when it comes to measurements below 200 pF or so, modern DMMs will produce coarse if not ridiculous results. Elektor's purpose-designed Pico C does a far better job. Beating many DMMs hands down, this little instrument easily and accurately measures capacitances down to fractions of a picofarad!

Kit of parts incl. Elektor Project Case, programmed microcontroller, LCD and PCB

Art.# 100823-71 • $118.40

Wireless OBD-II

(April 2011)

The cheapest way to diagnose faults on a modern car is to connect its OBD-II interface to a (notebook) PC running suitable diagnostics software. However, a wired connection is not always the most suitable, and selfcontained OBD testers are a rather expensive and less flexible alternative to using a PC. An interesting option is a wireless OBD interface with a radio interface to a PC: this homebrew solution allows the choice of using either Bluetooth or ZigBee.

OBD2- Zigbee or Bluetooth interface kit with all parts and enclosure

Art.# 100872-71 • $201.70 (Zigbee)
Art.# 100872-72 • $201.70 (Bluetooth)

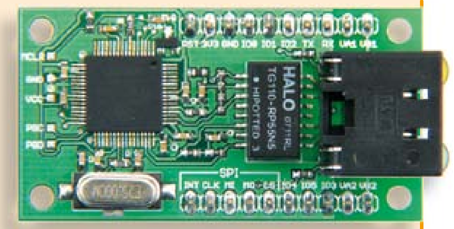
SatFinder

(March 2011)

Those of you who regularly need to realign a satellite TV dish will find this gadget extremely valuable. Caravan owners and campers on long journeys who crave their home TV channels can now keep up with developments in sports, news and the soaps back home with the help of the SatFinder. This GPS based design includes a database containing positional information of a number of popular TV satellites. With the help of GPS data it calculates the precise angles to find the satellite first time!

Kit of parts including Controller, Display and PCB (North American Version)

Art.# 100699-72 • $114.90



NetWorker

(December 2010)

An Internet connection would be a valuable addition to many projects, but often designers are put off by the complexities involved. The 'NetWorker', which consists of a small printed circuit board, a free software library and a ready-to-use microcontroller-based web server, solves these problems and allows beginners to add Internet connectivity to their projects. More experienced users will benefit from features such as SPI communications, power over Ethernet (PoE) and more.

Module, ready assembled and tested

Art.# 100552-91 • $85.50

Product Shortlist

June 2011 (No. 30) $

+ + + Product Shortlist June: See www.elektor.com + + +

May 2011 (No. 29)

Microphone Conferencing System

100465-1 ...... Printed circuit board...14.30

Elektor Proton Robot

110263-71 .... Kit of parts All Inclusive (Body + Head + Audio + Gripper + PIC Add-on)........................1745.00
110263-72 .... Kit of parts All Inclusive (Body + Head + Audio + Gripper + AVR Add-on) 1745.00
110263-78 .... Ready assembled and tested, PIC Add-on.................................55.00
110263-79 .... Ready assembled and tested, AVR Add-on 55.00
110263-91 .... Ready assembled and tested, PIC version 2375.00
110263-92 .... Ready assembled and tested, AVR version.............................2375.00

1-Channel DMX512 Light Dimmer

EB006........... E-block PIC MultiProgrammer 121.00
TEFLCST4....... E-block Flowcode 4 for PICmicro.................................74.10

Mobile, Text, CallerID

071035-72 .... Relay board with all parts and relays.................................73.80
071035-91 .... PCB, partly populated, ATM18 Controller module 15.40
071035-92 .... PCB, partly populated ATM18 Test board 48.30
071035-93 .... LCD board, SMD populated incl. pinheaders.............................37.10
071035-95 .... Port extension board, SMD populated 26.80

April 2011 (No. 28)

Pico C

100823-1 ...... Printed circuit board...14.30
100823-41 .... Programmed controller ATTINY2313-20PU 14.30
100823-71 .... Kit of parts incl. Elektor Project Case, programmed microcontroller and PCB 118.40

Wireless OBD-II

100872-71 .... OBD2-Zigbee interface kit incl. Zigbee-USB stick, all parts and enclosure...201.70
100872-72 .... OBD2-Bluetooth interface kit with all parts and enclosure 201.70

Asteroids & E-Blocks

EB014........... Keypad (E-block)...29.10
EB058........... Color graphics display (E-block)...............................121.00
EB655SI4....... dsPIC bundle (E-block) 482.30
TEDSSI4......... Flowcode for dsPIC Pro 282.30

Guitar Input for Multi-Effects Unit

100923-1 ...... Printed circuit board...11.50

Altimeter for Micro-Rockets

100418-41 .... PIC16F88-E/SO (SOIC-18), programmed....................................14.30

GPIB-to-USB Converter

080068-91 .... Controller board, populated and tested 88.80

ATM18 Catches the RS-485 Bus

071035-72 .... Relay board with all parts and relays.................................59.60
071035-91 .... ATM18 Controller module 15.40
071035-92 .... ATM18 test board...48.30
071035-93 .... LCD board, SMD populated incl. pinheaders.............................37.10
071035-95 .... Port extension board, SMD populated 21.70
080213-71 .... TTL-232R 5V cable...28.30

March 2011 (No. 27)

SatFinder

100699-1 ...... Printed circuit board...18.60
100699-42 .... ATMEGA8A-PU, programmed, US version.....................................14.20
100699-72 .... Kit of parts, US version...114.90

Mini Webserver using BASCOM-AVR

090773-91 .... Minimod18 Module 90.40

A String of 160 RGB LEDs

100743-1 ...... Printed circuit board...18.60
071035-91 .... PCB, partly populated, ATM18 Controller module 15.40
071035-92 .... PCB, partly populated ATM18-Testboard 48.30
071035-93 .... SMD-populated board with all parts and pinheaders...37.10

Bestsellers

Books

1 Design your own **Embedded Linux Control Centre on a PC**
ISBN 978-1-907920-02-8........................ $55.70

2 **Assembly Language Essentials**
ISBN 978-0-9630133-2-3........................ $47.60

3 **Introduction to Control Engineering**
ISBN 978-0-905705-99-6........................ $44.40

4 **C# 2010 Programming and PC interfacing**
ISBN 978-0-905705-95-8........................ $47.60

5 **Analogue Video**
ISBN 978-0-905705-96-5........................ $42.80

CD/DVD-ROMs

1 DVD **Elektor 2010**
ISBN 978-90-5381-267-9........................ $37.90

2 CD **Elektor's Components Database 6**
ISBN 978-90-5381-258-7........................ $40.20

3 CD **ATM18 Collection**
ISBN 978-0-905705-92-7........................ $39.60

4 DVD **Wireless Toolbox**
ISBN 978-90-5381-268-6........................ $46.00

5 DVD **Elektor 1990 through 1999**
ISBN 978-0-905705-76-7........................ $111.30

Kits & Modules

1 **Pico C Meter**
Art. #100823-71 .... $118.40

2 **NetWorker**
Art. # 100552-91.. 85.50

3 **Wireless OBD-II**
Art. # 100872-71/72............................ $201.70

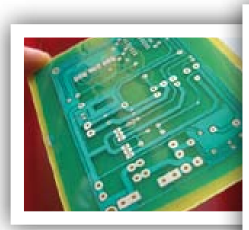
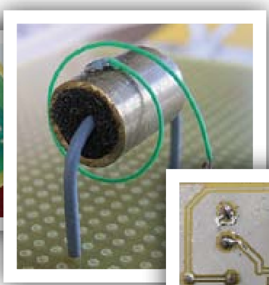
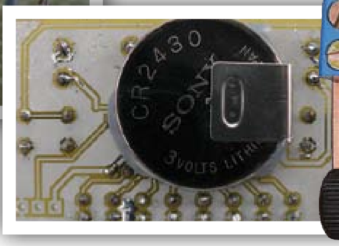
4 **SatFinder**
Art. #100699-71 .... $114.90

5 **MIAC-PLC**
Art. # MIO235 .... $248.40

Order quickly and securely through
www.elektor.com/shop
or use the Order Form near the end of the magazine!

elektor

Elektor US
PO Box 180
Vernon, CT 06066
USA
Phone: 860-875-2199
Fax: 860-871-0411
E-mail: order@elektor.com

Project Generator Edition 2011

Elektor's annual load of small circuits, ideas and tips for the electronics & embedded communities

No one does it better! In Elektor's July & August 2011 double edition the editors, designers and several enthusiastic contributors again launch a widely varied collection of electronics-related articles covering cute projects, new IC applications, junkbox explorations, experimental circuits and sub-circuits, software and design tips.

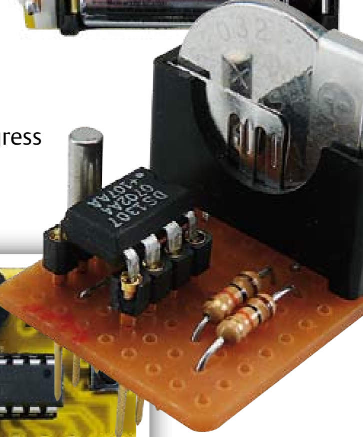
Do not miss our best selling edition of the year!

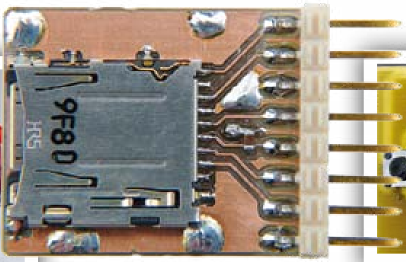
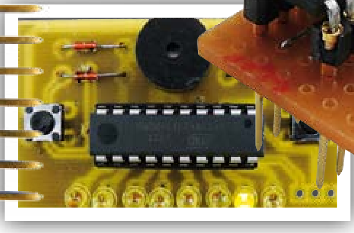
A selection from the contents:

Movement Sensor	Serial AVR Programmer	Electronic Dog Whistle
Dip Meter	Backup Power Supply	Running Lights
Tandem Doorbell	MultiFlasher	Ring Oscillator
Chaos Generator	Universal Component Tester	DC-DC Converter
Audion Receiver	Water Detector	Floating Supply

Extra – Extra – Extra!

The July & August edition traditionally contains one larger project spanning several pages. This project is in progress at the time of writing and we hope you will be pleasantly surprised. All will be revealed in a few weeks time!



Article titles and magazine contents subject to change; please check the Magazine tab on www.elektor.com

Elektor UK/European July & August 2011 edition: on sale June 23, 2011.

Elektor USA July & August 2011 edition: published June 20, 2011.

w.elektor.com www.elektor.com www.elektor.com www.elektor.com www.elektor.com ww

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

News
New IP core accelerates dataplane and DSP applications
Mobile phones and laptops charge from the sun
Wristwatch computer with 3.5 inch TFT screen
It's here! Elektor's Components Database 6 CD-ROM
Electrical wave blaster could provide new way to extinguish fires
News Overview

NEW: Elektor Academy
Distance Learning Course
Now at a special introductory price!
£40 / $70 / € 50 DISCOUNT
Programming Embedded PIC Microcontrollers

Subscribers only
NEW BOOK
13% Discount and Free Shipping for Elektor magazine subscribers!
Assembly Language Essentials

Elektor Magazine
Subscribe now to the greatest magazine in the field of electronics and information technology!

Subscribe now
Browse online

Subscribe to our free E-weekly Newsletter!
Enter your Email address:

Elektor PCB Prototyper
This compact, professional PCB

E-blocks & Flowcode
The gateway to µC programming and technology

App for iPhone and iPad
Elektor Electronic Toolbox
A must have for electronics enthusiasts!

Optimizing C Compiler for Microchip PIC® MCU
sensors expo & conference
JUNE 6-8, 2011 ROSEMONT, IL

Also on the Elektor website:

- Electronics news and Elektor announcements
- Readers Forum
- PCB, software and e-magazine downloads
- Time limited offers
- FAQ, Author Guidelines and Contact

Order Form 06-2011

Description	Price each	Qty.	Total	Order Code
Linux – PC-based Measurement Electronics NEW	$47.60			
Assembly Language Essentials NEW	$47.60			
Design your own Embedded Linux Control Centre on a PC NEW	$55.70			
CD Elektor's Components Database 6 NEW	$40.20			
CD ATM18 Collection	$39.60			
DVD Elektor 2010	$37.90			
Introduction to Control Engineering	$44.40			
Sub-total				
Shipping & Handling *(Airmail from Europe)*	$		5.00	
Total paid				

Prices and item descriptions subject to change. The publishers reserve the right to change prices without prior notification. Prices and item descriptions shown here supersede those in previous issues. E. & O.E.

Name/Company

Street & No.

City/State/Zip *Country*

E-mail *Tel*

EL06 *Date* – – *Signature*

METHOD OF PAYMENT

Remit in US $ drawn on a US bank. Please see next page for ordering information.

- ☐ Check or Money Order
- ☐ Mastercard
- ☐ VISA
- ☐ Discover
- ☐ American Express

Expiry date: ____________________

Verification code: _______________

Send order form to:

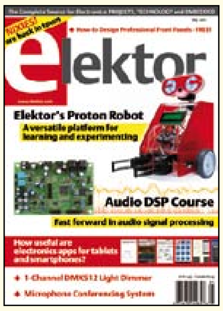
Elektor US
PO Box 180
Vernon, CT 06066
USA

Phone: 860-875-2199
Fax: 860-871-0411
E-mail: order@elektor.com
www.elektor.com/shop

elektor

Subscription Order Form 06-2011

Yes, I want to subscribe to Elektor US for 1 year*



I would like:

- ☐ **Standard Subscription for $39.95 (11 issues)**
- ☐ **Plus Subscription for $59.95 (11 issues + the Elektor Volume 2011 DVD-ROM + exclusive access to www.elektor-plus-usa.com)****

Name/Company

Street & No.

City/State/Zip *Country*

E-mail *Tel*

EL06 *Date* – – *Signature*

* *Offer available in US and Canada only. Canada please add $11.00 per year for postage.*
** *Please note: For a Plus subscription, it's important that you fill in your email address. This will enable you to have access to Elektor-plus.usa.com where you can browse an online Elektor archive exclusively available to Elektor Plus subscribers!*

METHOD OF PAYMENT

Remit in US $ drawn on a US bank. Please see next page for ordering information.

- ☐ Check or Money Order
- ☐ Mastercard
- ☐ VISA
- ☐ Discover
- ☐ American Express

Expiry date: ____________________

Verification code: _______________

Send order form to:

Elektor US
PO Box 180
Vernon, CT 06066
USA

Phone: 860-875-2199
Fax: 860-871-0411
E-mail: service@elektor.com
www.elektor.com/subs-usa

ORDERING INFORMATION

To order contact customer service:

Phone:	860-875-2199
Fax:	860-871-0411
Mail:	Elektor US PO Box 180 Vernon, CT 06066 USA
E-mail:	sales@elektor.com
On-line at	www.elektor.com

Customer service hours: 8:00 AM–4:30 PM Monday–Thursday. Voice mail available at other times.
When leaving a message please be sure to leave a daytime telephone number where we can return your call.

PLEASE NOTE: While we strive to provide the best possible information in this issue, pricing and availability are subject to change without notice. To find out about current pricing and stock, please call or email customer service.

COMPONENTS

Components for projects appearing in Elektor are usually available from certain advertisers in the magazine. If difficulties in obtaining components are suspected, a source will normally be identified in the article. Please note, however, that the source(s) given is (are) not exclusive.

PAYMENT

Orders must be prepaid. We accept checks or money orders (in US $ drawn on a US bank only), VISA, Mastercard, Discover, and American Express credit cards. We do not accept C.O.D. orders.
We also accept wire transfers. Add $20 to cover fees charged for these transfers.

TERMS OF BUSINESS

Shipping Note: All orders will be shipped from Europe. Please allow 3–4 weeks for delivery. Shipping and handling via airmail: US $20.00 per order. **Returns** Damaged or miss-shipped goods may be returned for replacement or refund. All returns must have an RA #. Call or email customer service to receive an RA# before returning the merchandise and be sure to put the RA# on the outside of the package. Please save shipping materials for possible carrier inspection. Requests for RA# must be received 30 days from invoice. **Patents** Patent protection may exist with respect to circuits, devices, components, and items described in our books and magazines. Elektor accepts no responsibility or liability for failing to identify such patent or other protection. **Copyright** All drawing, photographs, articles, printed circuit boards, programmed integrated circuits, diskettes, and software carriers published in our books and magazines (other than in third-party advertisements) are copyrighted and may not be reproduced (or stored in any sort of retrieval system) without written permission from Elektor. Notwithstanding, printed circuit boards may be produced for private and personal use without prior permission. **Limitation of liability** Elektor shall not be liable in contract, tort, or otherwise, for any loss or damage suffered by the purchaser whatsoever or howsoever arising out of, or in connection with, the supply of goods or services by Elektor other than to supply goods as described or, at the option of Elektor, to refund the purchaser any money paid with respect to the goods.

SUBSCRIPTIONS (US & CANADA ONLY)

Subscription rates (1 Yr.)

Standard Subscription:	$39.95
Plus Subscription:	$59.95

Canada add $11 per year for postage

All subscriptions begin with the current issue. Expect 3–4 weeks for receipt of the first issue. Subscriptions, renewals, and change of address should be sent to:

Elektor US
PO Box 180
Vernon, CT 06066
USA

E-mail: sales@elektor.com

Order subscriptions on-line at www.elektor.com/subs

Subscriptions may be paid for by check or money order (in US $ drawn on a US bank only). We accept Mastercard, VISA, Discover and American Express credit cards.

For gift subscriptions, please include gift recipient's name and address as well as your own, with remittance. A gift card will be sent on request.
Subscriptions may be cancelled at any time for a refund of all unmailed issues.

Does your subscription expire soon?
Renew it online at www.elektor.com/renew

Elektor App for iPhone and iPad

The Elektor Electronic Toolbox

Elektor now offers an App you cannot afford to miss on your iPhone, iPod Touch or iPad. The Elektor Electronic Toolbox is a collection of no fewer than 28 electronic tools that can be picked from a comprehensive set of icons!

Highlights:

- Databases for Transistors, FETs, Triacs, Thyristors, Diodes and ICs
- NE555 circuit design
- An Ohms Law calculator
- Schematic Diagrams
- Number base converter
- LED / resistor calculation
- R/L and BJT calculations
- And more

Now available from the Apple iTunes Store for just $5.99!

Further information at

www.elektor.com/app

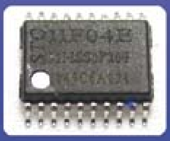
STC Microcontroller
The World's Largest 8051 MCU Family

- Enhanced 80C51 CPU, 1T per machine cycle
- Binary level compatible with conventional 8051
- 0.5-62K on-chip flash ROM
- Up to 2048KB SRAM
- In-System program or In-Application program
- Internal oscillator, reset, WDT
- On-chip ADC, PWM, EEPROM, SPI
- Addressing up to 64KB of external RAM
- Dual data pointer
- 6 vector address, 2 level priority interrupt
- Up to 3 UART with baud-rate generator
- Low power consumption
- 8 to 48Pins, DIP/PLCC/QFN/SOP/DIP/SSOP
- Ultra safe code protection for flash ROM
- Highly EDS protecting

- 1000+ parts
- Free ISP software
- Keil IDE supporting
- Instant order online
- World wide free shipping

Free USB ISP Tool

Feature Product



STC11F04E

Unit Price: **$0.86@1ku**

Compatible with AT89C4051
6–8 times faster in average
2K EEPROM
16, 18, 20pins DIP/SOP/LSSOP

Reduce the cost dramatically while considerably improving the performance

Tel: +8610 8256 2708 Email: sales@stc-51.com
Website: http://www.stc-51.com

Index of Advertisers

AP Circuits, Showcase	www.apcircuits.com	31, 78
Atria Technologies, Showcase	www.AtriaTechnologies.com	78
BusBoard, Showcase	www.BusBoard.us	78
Circuit Cellar	www.circuitcellar.com	27, 47, 79
Decade Engineering, Showcase	www.decadenet.com.	78
DLP Design	www.dlpdesign.com	31
ExpressPCB	www.expresspcb.com.	9
EzPCB	http://www.stc-51.com	87
Hameg Instruments, Showcase	www.hameg.us	78
Images Scientific, Showcase	www.imagesco.com	78
Linx Apex Wireless, Showcase	www.apexwireless.com.	78
Linx Technologies.	www.linxtechnologies.com	67
Maxbotix, Showcase	www.maxbotix.com	78
Mikroelektronika.	www.mikroe.com	3
NXP Product	www.nxp.com/microcontrollers.	88
Paia Electronics, Showcase	www.paia.com	73, 78
Parallax	www.parallax.com	25
Pololu Corporation, Showcase	www.pololu.com.	2, 78
Saelig	www.saelig.com	2
Showcase		78, 79
Tag-Connect, Showcase.	www.plugofnails.com	78
Triangle Research, Showcase.	www.tri-plc.com/ek.htm	78

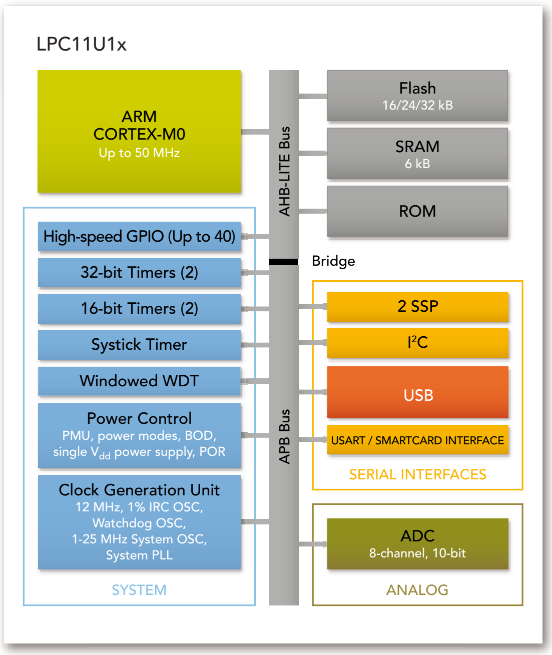
Advertising space for the issue 11 August 2011

may be reserved not later than 19 July 2011

with Strategic Media Marketing, Inc. - 2 Main Street - Gloucester, MA 01930 - USA Telephone 1.978.281.7708 - Fax 1.978.281.7706 - e-mail: ElektorUSA@smmarketing.us to whom all correspondence, copy instructions and artwork should be addressed.

LPC11U00
Low cost Cortex-M0 USB solution

- SmartCard interface
- Small form-factor mobile and consumer applications
- Migration path to Cortex-M3
- Better approach to USB with configurable buffer management
- Up to 32K Flash and 6K RAM (2K USB SRAM)


LPC11U1x
ARM CORTEX-M0 Up to 50 MHz
AHB-LITE Bus
Flash 16/24/32 kB
SRAM 6 kB
ROM
Bridge
High-speed GPIO (Up to 40)
32-bit Timers (2)
16-bit Timers (2)
Systick Timer
Windowed WDT
Power Control PMU, power modes, BOD, single V_{dd} power supply, POR
Clock Generation Unit 12 MHz, 1% IRC OSC, Watchdog OSC, 1-25 MHz System OSC, System PLL
SYSTEM
APB Bus
2 SSP
I^2C
USB
USART / SMARTCARD INTERFACE
SERIAL INTERFACES
ADC 8-channel, 10-bit
ANALOG

www.nxp.com/microcontrollers