



elektor

LEARN

DESIGN

SHARE

DDS Function Generator

Sines, squares and triangles
up to 10 MHz

In This Edition:
7 Labs Projects
3 Reader's Projects
6 New Modules & PCBs
4 Courses
2 Reviews
and lots more!

Add a Bill of Materials • An
Analog Robot • Android I/O Board

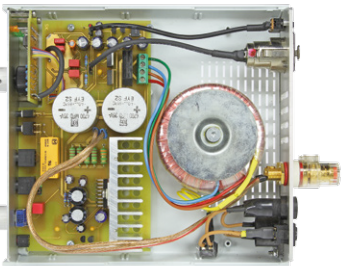
(2) • ARM CSIS Developer Competition

• ARM'ed T-Board • ARM Micros and the SPI Bus •

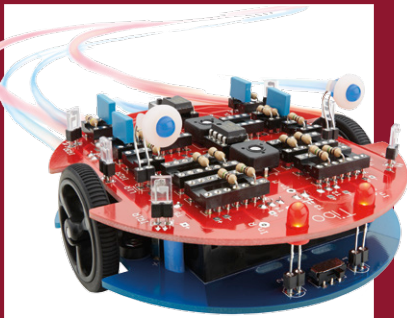
BL-600 e-BoB (5) • Bullheaded Buck Converter • Coax Connectors • Compact
60-watt Amplifier • Compact USB to Serial Converter • DDS Function Generator

• Digital Zoetrope • Dot Labs is Dee-I-Wye • e-ethics • Enhanced FM Stereo
on Red Pitaya • Hexadoku • LED Driver with High PF and Ultra Wide Output
Range • MEASSY • Not enough pins again? • PIC replacement for a 555 timer

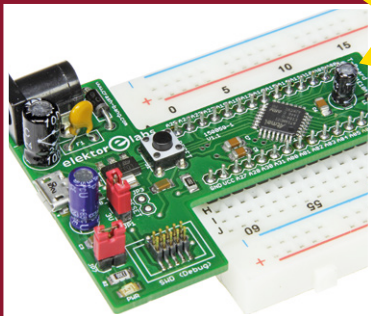
• Rise of the Drones • Rohde & Schwarz HMC8043 PSU • The Teflon Grooved-
Tape Recorder • Two Arms for the ARM • Windows on the Raspberry Pi



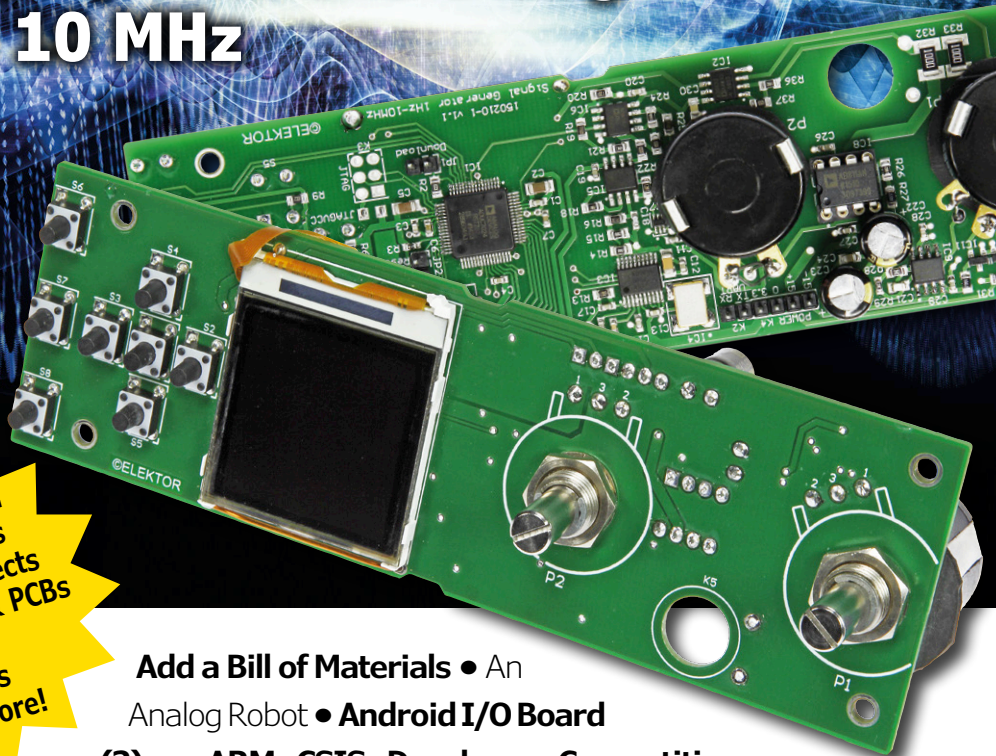
MEASSY
For audiophile
speaker builders



Analog Robot
Tibo, from concept to
construction

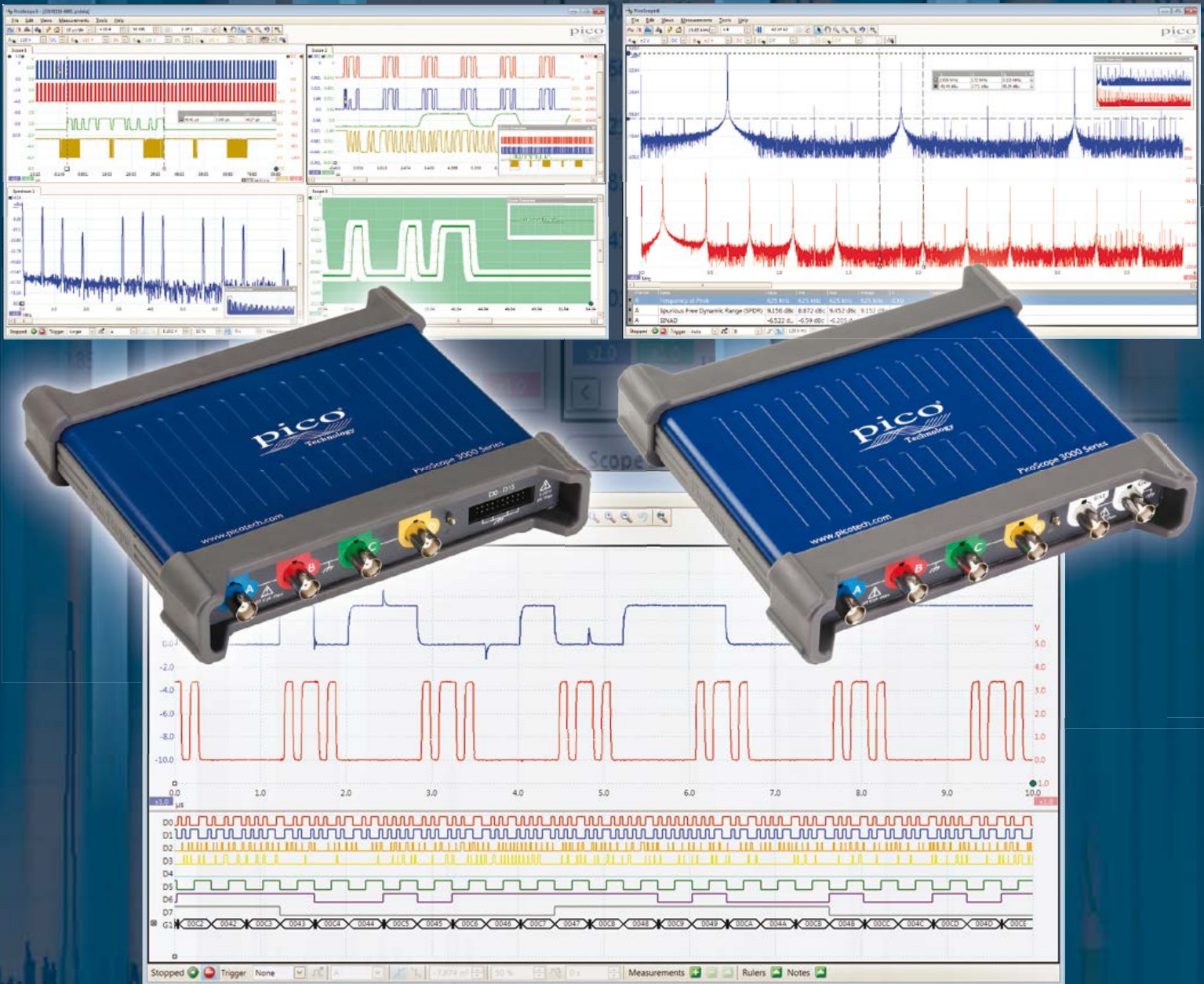


ARM'ed T-Board
Here's Cortex M0+
32-bit power



THE PICOSCOPE 3000 SERIES

- Up to 200 MHz analog bandwidth
- Deep buffer memory up to 512 MS
- MSO models with 16 digital channels
- 2 or 4 analog channels
- 1 GS/s real-time sampling



Includes serial bus decoding and analysis (CAN, LIN, RS232, I2C, I2S, SPI, FlexRay), segmented memory, mask testing, spectrum analysis, and software development kit (SDK) all as standard, with free software updates and five years warranty.

Edition 6/2015
Volume 41, No. 467 & 468
November & December 2015

ISSN 1947-3753 (USA / Canada distribution)
ISSN 1757-0875 (UK / ROW distribution)
www.elektor.com,
www.elektormagazine.com

Elektor Magazine, English edition
is published 6 times a year by

Elektor International Media
78 York Street
London W1H 1DP, UK
Phone: (+44) (0)20 7692 8344

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

USA / Canada Memberships:
Elektor USA
P.O. Box 462228
Escondido, CA 92046
Phone: 800-269-6301
E-mail: elektor@pcspublink.com
Internet: www.elektor.com/member

UK / ROW Memberships:
Please use London address
E-mail: service@elektor.com
Internet: www.elektor.com/member

Advertising & Sponsoring:
Johan Dijk
Phone: +31 6 15894245
E-mail: johan.dijk@eimworld.com
www.elektor.com/advertising
Advertising rates and terms available on request.

Copyright Notice
The circuits described in this magazine are for domestic and educational use only. All drawings, photographs, printed circuit board layouts, programmed integrated circuits, disks, CD-ROMs, DVDs, software carriers, and article texts published in our books and magazines (other than third-party advertisements) are copyright Elektor International Media b.v. and may not be reproduced or transmitted in any form or by any means, including photocopying, scanning and recording, in whole or in part without prior written permission from the Publisher. Such written permission must also be obtained before any part of this publication is stored in a retrieval system of any nature. Patent protection may exist in respect of circuits, devices, components etc. described in this magazine. The Publisher does not accept responsibility for failing to identify such patent(s) or other protection. The Publisher disclaims any responsibility for the safe and proper function of reader-assembled projects based upon or from schematics, descriptions or information published in or in relation with Elektor magazine.

© Elektor International Media b.v. 2015
Printed in the USA Printed in the Netherlands



Up for rectification



After some statistical research applied to my IN boxes, one 2 GB (editor@elektor.com) and one steel (downstairs@reception) I found that after rectification and ripple suppression, 70.71 percent ($\frac{1}{2}\sqrt{2}$) of writing Elektor Magazine readers suffer from the misconception that the contents of their journal originates from a mysterious, closed circle of authors. At shows too, I get responses of wonder, surprise and enthusiasm from engineers young and old being told that “from issue one back in 1975, Elektor has been totally open to contributions from the audience”. Theirs too; yours too. There are two ways to getting a project published in Elektor Magazine; I shall print them in this space.

- 1. LABS PROJECT.** Sign up to elektor-labs.com and propose it there in outline. If your project is selected by Elektor Labs staff, it will enter the ‘In Progress’ phase. Subsequently it gets post engineered to meet Elektor standards, and proposed to the international editorial team for text/graphics processing and publication. Post engineering includes (re)producing a PCB design, circuit and component scrutiny, and optionally enhancement, by an Elektor Labs engineer. Optionally & to be discussed with you: video, product support and sales of the item in the Elektor Store. Importantly, publishing on elektor-labs.com does not warrant publication in Elektor Magazine.
- 2. READER'S PROJECT.** Propose it to me as a fully worked out manuscript, complete with images. Your proposal gets evaluated by a 3-person committee. I am not (yet) on the committee. If accepted, the material gets published as a Reader's Project. No labs support is available and the project gets published with the necessary editing and formatting to meet Elektor graphics and presentation standards. Schematics will be redrawn in Elektor style.

Both 1. and 2. qualify for payment to be negotiated, and an author contract. Also, for publication in French, German and Dutch.

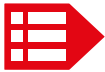
All firmware associated with the project must be open source and made available for free downloading by your fellow Elektor Members down to the source code level. Please review the DESIGN section in this copy of Elektor magazine for fine examples of READER'S PROJECT and LABS PROJECT.

Looking forward to your contribution (raw DC okay)

Jan Buiting, Editor-in-Chief

The Circuit

Editor-in-Chief:	Jan Buiting
Publisher:	Don Akkermans
Membership Manager:	Raoul Morreau
Support Executive:	Cindy Tijssen
International Editorial Staff:	Thijs Beckers, Mariline Thiebaut-Brodier Denis Meyer, Jens Nickel
Laboratory Staff:	Ton Giesberts, Luc Lemmens, Clemens Valens, Jan Visser
Graphic Design & Prepress:	Giel Dols
Online Manager:	Daniëlle Mertens



THIS EDITION

Volume 41 – Edition 6/2015

No. 467 & 468 November & December 2015

- 6 Elektor Circuits & Connections
- 36 ElektorBusiness: News & New Products
- 38 ElektorBusiness: LED Driver with High PF and Ultra Wide Output Range
- 42 Welcome to Elektor Labs
- 104 ARM CSIS Developer Competition
- 106 Elektor Store
- 129 Elektor World News
Coming soon: e-ethics
- 130 Play & Win:
Hexadoku, the original Elektorized hexadoku

LEARN

DESIGN

SHARE

- 8 Welcome to the LEARN section
- 9 Readers Tips & Tricks
Not enough pins again?
- 10 From 8 to 32 Bits:
ARM Microcontrollers for Beginners (6)
The SPI Bus
- 16 DesignSpark Mechanical CAD Tips & Tricks (3)
Add a Bill of Materials
- 18 PIC Assembler Crash Course (3):
PIC replacement for a 555 timer
- 25 Peculiar Parts, the series
Coax Connectors
- 26 An Analog Robot
- 30 Windows on the Raspberry Pi (1)
Installation and first programs

LEARN

DESIGN

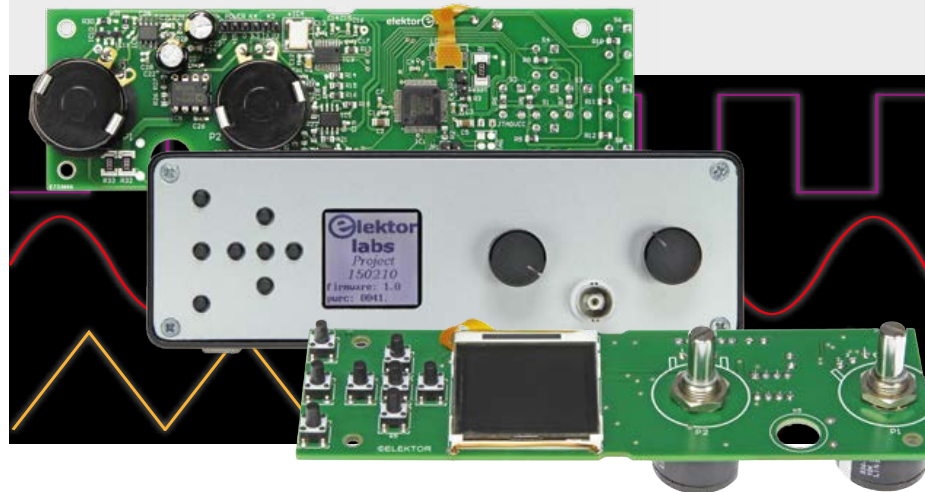
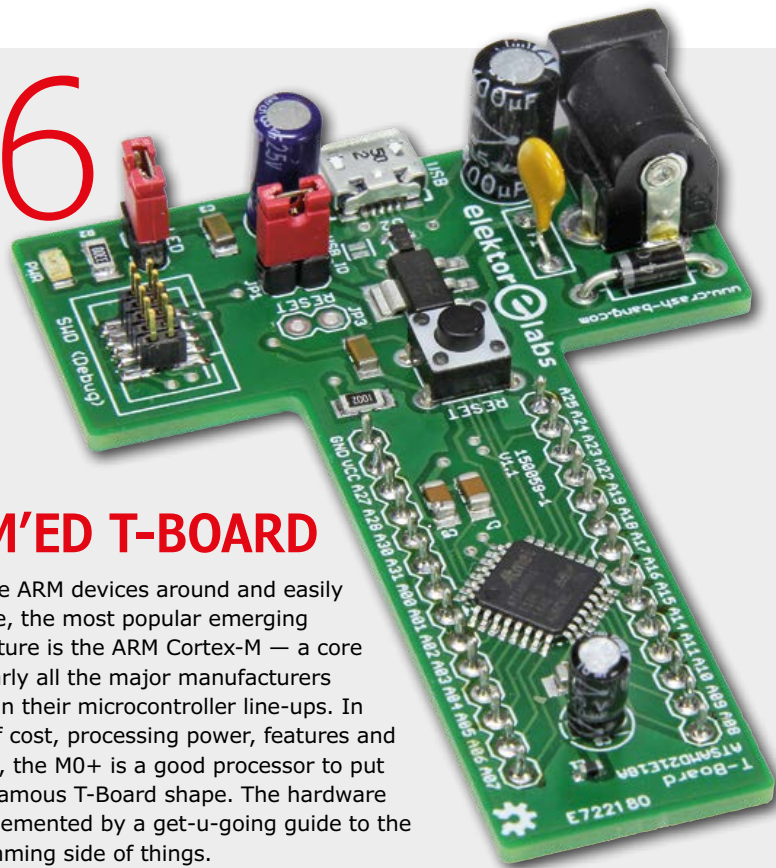
SHARE

- 44 Welcome to the DESIGN section
- 45 Compact USB to Serial Converter
- 48 Android I/O Board (2)

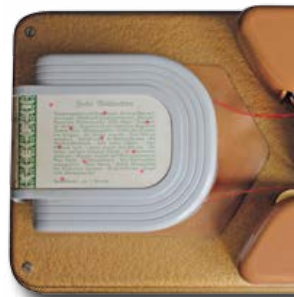
56

ARM'ED T-BOARD

Of all the ARM devices around and easily available, the most popular emerging architecture is the ARM Cortex-M — a core that nearly all the major manufacturers include in their microcontroller line-ups. In terms of cost, processing power, features and I/O pins, the M0+ is a good processor to put on our famous T-Board shape. The hardware is complemented by a get-u-going guide to the programming side of things.



- 56 ARM'ed T-Board
- 64 Compact 60-watt Amplifier
- 68 DDS Function Generator
- 76 BL-600 e-BoB (5)
- 82 MEASSY
- 90 Enhanced FM Stereo on Red Pitaya
- 94 Two Arms for the ARM
- 101 Digital Zoetrope



124

MEASSY

At some point, even the most passionate loudspeaker builder has enough of the mess and unreliability of improvised instrumentation setups. A cable here, an adapter there, and where did that wire go to? This frustration drove the author to develop a proper instrument. Introducing MEASSY: the measuring instrument for loudspeakers.



LEARN

DESIGN

SHARE

110 Welcome to the SHARE section

112 Review

Rohde & Schwarz HMC8043 PSU —
how does it stack up?

116 Web Scouting

Rise of the Drones

118 Escaped from the Labs

Bullheaded Buck Converter

120 What's Hot at dot Labs

Dot Labs is Dee-I-Wye

122 Retronics

The Teflon Grooved-Tape Recorder

DDS Function Generator

Sines, squares and triangles up to 10 MHz

68



NEXT EDITION

Active Crossover

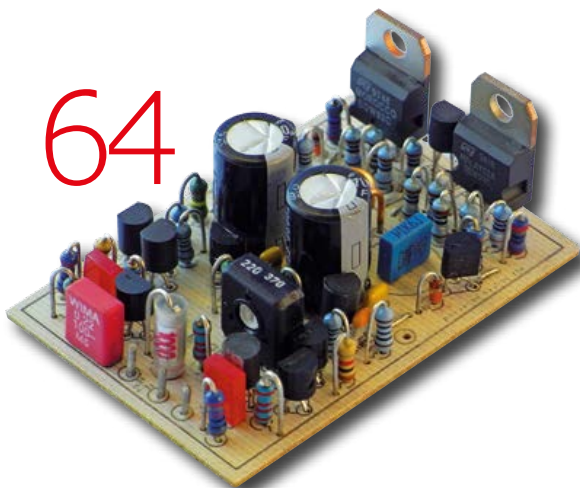
This active 3-way crossover filter is designed with versatility in mind. One possible application is an active speaker system.

Modular PSU

This power supply based on DC/DC-converters has floating outputs and can be configured in different ways, like 3.3 V/0.5 A or ± 15 V/0.33 A. It is conveniently powered by a 19-V notebook-power adapter.

Fridge Thermostat Adjuster with Android I/O Board

To an Elektor reader, replacing the fridge thermostat is a piece of cake. Finding the right thermostat though is a lot more difficult. Using an Android I/O board we make a fridge thermostat adjusting device that allows a cheap, general-purpose thermostat to be used.



Elektor Magazine edition 1 / 2016 covering January & February is published on January 5, 2016.

Delivery of printed copies to Gold members subject to transport.

Contents and article titles subject to change.

Elektor Circuits &

Elektor breaks the constraints of a magazine. It's a community of active e-engineers — from novices to professionals — eager to learn, make, design, and share surprising electronics.

57

Countries

246853

Enthusiastic Members

10

Experts &



Elektor Post

The e-inspiration weekly

Never monostable and with trigger signals all over the contents, Elektor's dot-Post weekly newsletter has the ability to ring in the weekend with gossip, techtalk, stray bits, previews and news flashes. And a project every other week.

www.elektor.com/newsletter



Elektor Community

Become a member, Green or Gold

Membership of the Elektor Community is the surest way to enjoy classic electronics and embedded technologies side by side, ranging from beginner to pro. With direct access to Elektor Labs, forums, discounts, weekly newsletters, biweekly online projects, article archives, search engines, and back articles Green and Gold Members have permanent priority seating. Go GREEN if you want the magazine front to back delivered online only, or GOLD for the sumptuous package including printed copies.

www.elektor.com/memberships



Elektor TV

We're tubed too

No film set, suits, or Action! but you can rely on a camera rolling whenever things start humming, booting, displaying or smoking at Labs, or indeed any place or event our presenters find video compatible. Check out elektor.tv.

www.youtube.com/user/ElektorIM



Elektor PCB Service

Boards at your service

Forget the chemicals, get your electronic project to work as expected by ordering a ready-manufactured circuit board. Fast turnaround, pure quality, worldwide shipping.

www.elektorpcbservice.com



Elektor Labs

Learn, Design & Share

The techno creative center of Elektor that's steeped in hard core electronics all the way from scribble to PCB, component and kit. Wide open and accessible through its own website, Labs is where projects large, small, analog, digital, new and old skool are sketched, built, discussed, debugged and fine-tuned for replication and use by you.

www.elektor-labs.com



Elektor Academy

Ride the learning curve

Webinars, seminars, courses, presentations, workshops, lectures, in-company trainings, DVDs, and demos are just a few of the methods Elektor is using to spread the word about electronics both at hobby and professional levels.

www.elektor-academy.com

Connections Guide

31

479

233628

14:17
OCTOBER 06 2015

Authors

Publications

Montly Visitors

Print Time



Elektor Magazine

Close to 1024 pages of surprising electronics a year

If you prefer to absorb electronics over being absorbed, stick to reading Elektor's flagship product DMA'ed to you by their international editorial team. Whether arriving online or on paper every magazine edition is packed with electronics all-sorts for you to enjoy and explore in your own time. Free! Sign up!

www.elektormagazine.com



Elektor Web Store

Fill your shopping cart

Elektor has confidence in the products and services generated by Labs, Magazine, and selected business partners. That's why a brightly illuminated online retail store is open 24/7/365, with ordering and payment facilities for clients all over the world. An Aladdin's Cave of electronic parts and gizmos.

www.elektor.com



Elektor Books & DVDs

E-Information Powerpacks

It's hard to find a field of electronics not covered in depth and with authority by the products in our book and DVD portfolio. From reference work to programming course, 8-bit to ARM, Antenna to Zener diode; it's all there.

www.elektor.com

Become a member today!

GOLD

€1.75 per week
£1.27 / US \$1.97

- ✓ Elektor Annual DVD
- ✓ 6x Elektor Magazine (Print)
- ✓ 6x Elektor Magazine (Digital)
- ✓ Access to Elektor Archive
- ✓ Access to Elektor.LABS
- ✓ 10% Discount in Elektor Store
- ✓ Elektor.POST Newsletter
- ✓ 25 Extra Elektor Projects
- ✓ Exclusive Offers

www.elektor.com/gold

GREEN

€1.31 per week
£0.97 / US \$1.49

- ✗ Elektor Annual DVD
- ✗ 6x Elektor Magazine (Print)
- ✓ 6x Elektor Magazine (Digital)
- ✓ Access to Elektor Archive
- ✓ Access to Elektor.LABS
- ✓ 10% Discount in Elektor Store
- ✓ Elektor.POST Newsletter
- ✓ 25 Extra Elektor Projects
- ✓ Exclusive Offers

www.elektor.com/green

FREE

- ✗ Elektor Annual DVD
- ✗ 6x Elektor Magazine (Print)
- ✗ 6x Elektor Magazine (Digital)
- ✗ Access to Elektor Archive
- ✗ Access to Elektor.LABS
- ✗ 10% Discount in Elektor Store
- ✓ Elektor.POST Newsletter
- ✓ 25 Extra Elektor Projects
- ✓ Exclusive Offers

www.elektor.com/newsletter

Welcome to the **LEARN** section



By **Jens Nickel**

You will probably already have read coverage of this year's Internationale Funkausstellung IFA. I must admit it's been a few years since I last attended and thinking back, my enduring memories are of wide open spaces, enormous display screens and eager young besuited salespersons with perfect smiles who were not always able to answer all my technical questions. I thought there would probably not be enough of interest on offer this year to warrant an editorial piece. That at least was what I was thinking as I packed my bag before setting off to Berlin to attend this year's exhibition.

Once again, I was about to have my preconceptions overturned. Time for another learning experience. Along with the domestic appliances including coffee machines, vacuum cleaners and barbecue grills (all 'intelligent' of course) from the big names which were being demonstrated in the main halls, I found, tucked away in Hall 11.1 a place with a completely different atmosphere. One full day alone really wasn't enough time to explore all the latest electronic gizmos on show from the young start-up companies showing their wares there. It was worth the trip just for this one hall alone. Would you like to make electronic equipment completely waterproof? How about using your iPhone as a submersible camera? No problem, the start-up HZO would like to introduce you to a service they can provide. On their stand can see a large aquarium in which was submerged a functioning Raspberry Pi an iPhone and sundry other PCBs.

The process the company offers involves removing the electronic innards from electronic equipment and coating them with a thin waterproof film before reassembly. For a one-off individual treatment, the costs work out quite high. The company is looking for manufacturing partners where the process can be integrated into the product assembly stage to benefit from reduced costs of volume production (www.hzo.com).



The 'Panono' camera system is the answer if you are fed up with poor results from panorama picture apps on your smart phone. The Panono consists of a plastic ball the size of a grapefruit with 36 cameras recessed around its surface. Throw it into the air and all the cameras simultaneously snap a picture at the apogee of its travel. The Berlin-based startup (www.panono.com) says the system has generated much interest. Images from the cameras are electronically stitched together

to produce a 108 megapixel 360 degree picture which can be viewed on a PC, tablet or a Virtual Reality headset.

These bulky 3D headsets were everywhere at the IFA. The Fraunhofer Institute for Integrated Circuits IIS (developers of the MP3 standard) has developed an optimized playback system for digital music capable of producing immersive surround-sound effects from simple speakers and headsets (www.fraunhofer-cingo.com). The sound sources (placed anywhere in space) are capable of making you feel immersed

in the action happening on screen, even as you move your head. A virtual reality movie was demonstrated with sound produced by the surround sound speaker system. I mean seriously, could things get any cooler for a home cinema buff like me?

You must already be familiar with TV apps that allow use of a Tablet as a 'second screen' for TV viewing. Now the Ultra-HD-Zoom-App from Fraunhofer HHI allows you to follow a nominated player (in this case Bastian Schweinsteiger) while watching a televised football match. You get a close-up, of all his contributions while viewing the normal wide-angle view of the match on your TV. Now when it comes to the 'third half' and its time to crack open the beers and analyze your player's performance, you really can make authoritative contributions to the discussion based on your close-up view.

Educational

At the combined IFA stand for the German associations VDE, ZVEI and ZVEH there were also some special activities to encourage newcomers to electronics: a model of a fully automated house, a soldering workshop and a demonstration of robots built with Lego. (www.zvei.de). ◀

(150475)

Tips & Tricks

From and for readers

Here's another clever idea to make life easier for electronics enthusiasts.

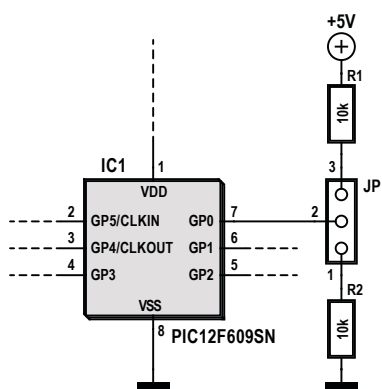


Not enough pins again?

By Dr. Martin Oppermann



Jumpers are often used to configure microcontroller circuits. That's easy if there are only two possible states to be configured – the desired state can be determined by fitting or not fitting a jumper on a port pin. However, if you have to distinguish between three possible states, you normally need an extra pin. Many years ago a Microchip application engineer gave me a tip that in any case works with PIC microcontrollers: the three settings are defined by the High, Low and Open states of a single pin (see the figure).



Use the following procedure to poll the port pin:

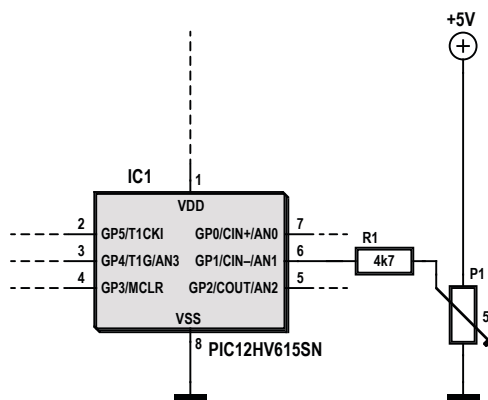
- Configure the pin as an output.
- Write a 1 to the pin.
- Configure the pin as an input.
- Read the pin and save the value.
- Configure the pin as an output.
- Write a 0 to the pin.
- Configure the pin as an input.
- Read the pin and add the value to the previously saved value.

If the result of the addition is 0 (0 + 0), the pin is Low. If the result of the addition is 2 (1 + 1), the pin is High. If the result of the addition is 1 (1 + 0), the pin is open (floating), which corresponds

to the third state. Note: to avoid excessive current consumption, an open pin should be quickly switched back to an output and set to a defined level.

All this is made possible by the parasitic capacitances present in every circuit, the high input impedance of microcontroller inputs, and the small number of commands necessary for the polling procedure. The code finishes execution before the parasitic capacitances can be discharged.

You can even use this method with an analog input. I took advantage of this in a temperature control circuit (see figure). If a potentiometer is connected to the analog port, the target temperature is set by the potentiometer, but if pin GP1 is open (P1 is not fitted), then the target temperature is set to a fixed value.



Pin GP1 is polled the same way as in the previous example. If a potentiometer is connected, a DC voltage between 0 and 5 V will be present on pin GP1 (depending on the potentiometer setting), which is declared as a digital I/O for the polling procedure. The Schmitt trigger characteristic of the digital inputs ensures that the read value is always either High or Low. The result of adding the two values read from the pin will only be 1 if the input is open ◀

(150479)

Got a neat solution for a tricky problem? Using components or tools in ways they were never intended to be used? Think your idea to solve a problem is better than the usual method? Have you discovered a work-around that you want to share with us and fellow makers? Don't hang around; write to us now, for every tip we publish you'll earn 40 pounds!



The SPI bus

From 8 to 32 bits: ARM Microcontrollers for Beginners (6)

By Viacheslav Gromov (Germany)

As we have seen previously in this series, the SERCOM module in the SAM D20 can be configured as a U(S)ART or as an I²C interface. To complete our tour of the module, one interface remains to be described: SPI. In this installment we look at a large-scale project that uses the SPI bus as well as other already-familiar peripheral modules.

In the previous installment of this series we explored using the ADC with an LM335 temperature sensor. We now want to use that sensor in a larger-scale project which also uses the SPI bus: a temperature logger. We will use an EEPROM from the 25LC or 25AA families with an SPI connection: these devices are available in a wide range of storage capacities and are easy to drive. EEPROMs are often handy devices to have around in any case, as very large projects can benefit from a large amount of external storage. We also need an accurate time reference, preferably one that records hours and minutes. And as luck would have it, we have already seen how to use the RTC in calendar mode.

SERCOM module as SPI peripheral

Configuring the SERCOM module of the SAM D20 in SPI mode gives us an overall structure that is rather similar to that when it is used in I²C mode. The block diagram in **Figure 1** shows the SERCOM module on the left configured as an SPI master and on the right as an SPI slave, with the four SPI signals shown in the middle. Both configurations use a shift register

to receive or transmit the individual data bits, and there are accompanying data registers: TxDATA, to which the CPU can write bytes to be transmitted, and RxDATA, from which the CPU can read data bytes that have been received. The RxDATA register is also equipped with an additional buffer register to help ensure smooth data transfer.

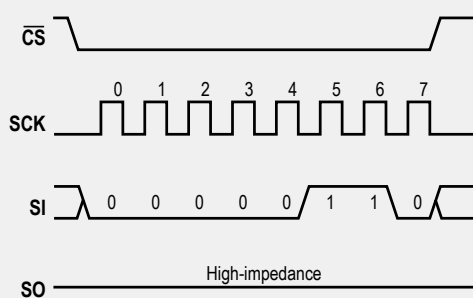
The SPI master also has responsibility for generating the bus clock: in the case of the SAM D20 this job is done by the baud rate generator. The clock frequency is set using the BAUD register. When configured as an SPI slave the module compares the received address (if present) with the values in ADDR and ADDRMASK to determine whether the microcontroller is the intended recipient of a data packet.

There are of course many details to the configuration, such as whether data are sent LSB first or MSB first. As a slave the SERCOM module can also operate in sleep mode (even slaves are allowed to sleep!). As usual with the SERCOM module, there is a choice of pins that can be used and there is also a choice of clock sources. More details can be found in the datasheet on page 369 [1].

The SPI protocol

'SPI' stands for Serial Peripheral Interface and, like I²C, it is a popular bus with a simple protocol [5]. Unlike I²C, however, SPI uses separate signals to carry data from and to the master. By not transmitting a slave address and by dispensing with start and stop bits, it achieves a very high data transfer rate, and the interface is therefore used in applications such as SD cards and in most ISP programming systems. The protocol allows as many slaves as required to be connected to a master using a single bus. SPI uses the following signals:

- **Master Out Slave In (MOSI):** data from master to slave
- **Master In Slave Out (MISO):** data from slave to master
- **Serial Clock (SCK):** clock to synchronize the bus, generated by the master
- **Slave Select (SS):** each slave has its own SS signal from the master. When the master takes the SS line low the slave knows that it is being addressed.



Source: Microchip

In contrast to the I²C bus, no pull-up resistors are used. Care needs to be taken to avoid problems in systems with mixed supply voltages (such as 3.3 V and 5 V). In the SPI protocol the relationship between the data signal and the clock signal is very important.

Both the polarity of the clock signal (CPOL) and its phase relative to the data signal (CPHA, which determines whether a data bit is valid on the falling or the rising clock edge) matter. The possible combinations of these settings give rise to four modes, numbered from 0 to 3. It is also possible for data to be transmitted LSB-first or MSB-first. How the interface is configured varies wildly from device to device, and there is no alternative but to resort to the datasheet for help.

The figure shows a typical SPI transfer: this particular example is a WREN command being sent to the 25LC040.

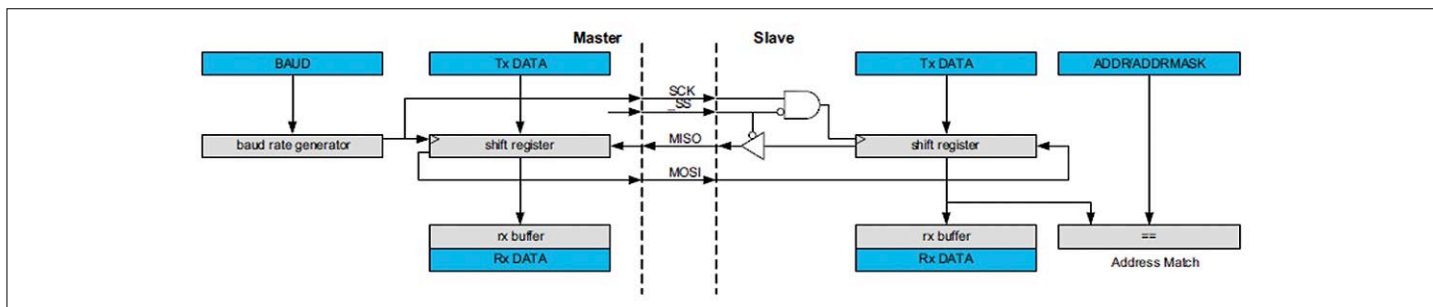


Figure 1. Outline block diagram of the SERCOM module configured as an SPI peripheral in slave and master modes (block diagrams and screenshots courtesy Atmel).

The temperature logger

It is a good idea to start with a high-level view of the circuit and the various peripheral modules and other components that will be used in it. Since the microcontroller's data sheet gives a list of all the peripheral modules and the pins to which they can be connected, it is easy to work out where the external components should be attached. There are of course many ways this can be done, as in many cases the device includes several identical peripheral modules (such as SERCOM1 to SERCOM4) and their pins are multiplexed to allow a choice of connections. To start with, however, we will use the pins in their standard configuration.

Now it is an easy matter to draw the circuit. For this project we will connect the LM335 using the same circuit as in the previous installment in this series, and we will attach a 25LC040 EEPROM using a minimal circuit: see **Figure 2**. The circuit can easily be built on a breadboard using 'Dupont' cables (**Figure 3**): as usual with such designs it is a good idea to keep the connections short.

Next to the software. First we create an empty project and bring in the required ASF libraries using the ASF wizard. If in doubt, include a library rather than omitting it: basic libraries such as those providing delays or interfacing to the U(S)ART often come in handy, even if only when debugging the project. Any superfluous libraries can always be removed later on. In this case (see **Figure 4**) we need a large number of libraries, the most important being those supporting the U(S)ART, the SPI bus and the RTC. Apart from the RTC library, all are configured in their polled versions, which means they do not generate interrupts, and this simplifies the main part of the code. In accordance with standard practice, we reuse the configuration functions for the UART, ADC, RTC and SPI modules from previous projects, and likewise we copy the callback interrupt service routine (ISR) for the RTC. Pin PA04 (AREFB) is already in use by the EEPROM and so we must change the code in the ADC configuration function taken from the earlier project so that it uses GPIO PA03 (AREFA) as its reference instead.

A new element is the SPI master configuration function for the SERCOM module, which we will look at in more detail: see **Listing 1** and [3]. First two configuration structures are declared, `config_spi_master` and `slave_dev_config`. The latter is then populated with its default settings. Then the command

```
slave_dev_config.ss_pin = PIN_PA05;
```

is used to tell the microcontroller to use pin PA05 for the SPI slave select (SS) signal. The settings in `slave_dev_config` are

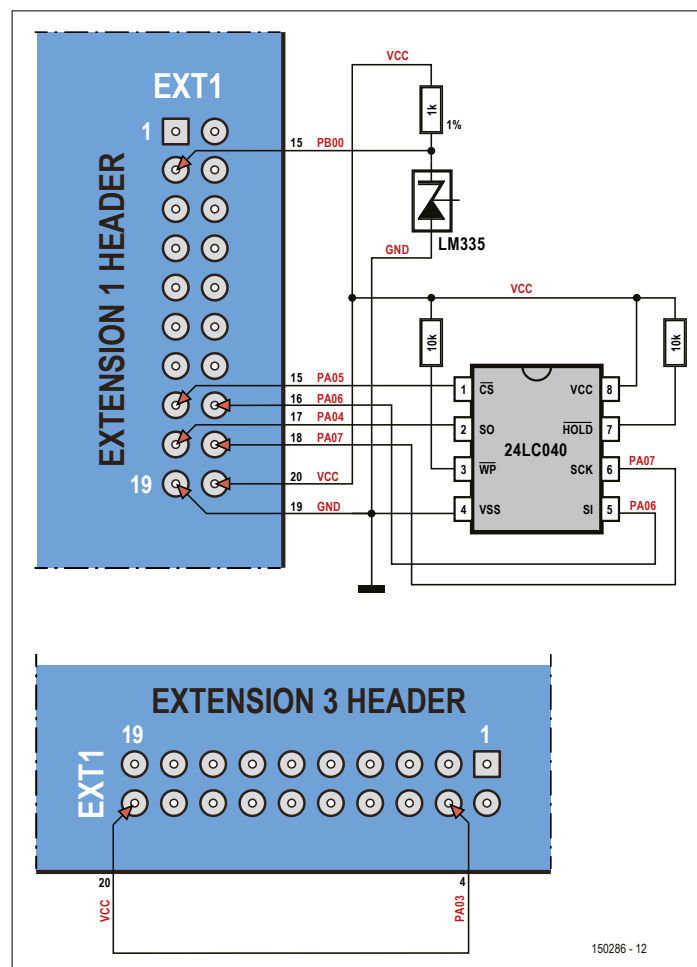


Figure 2. The simple temperature logger circuit.

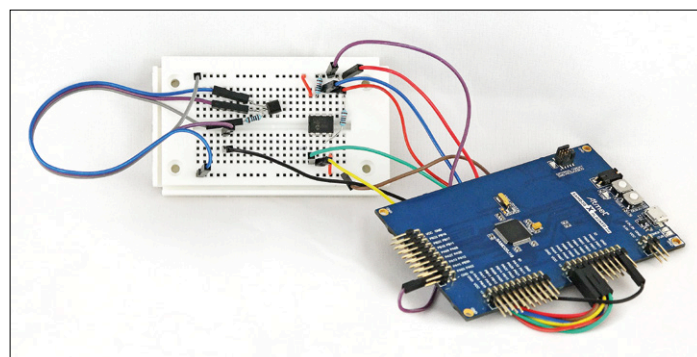


Figure 3. The author's prototype built on a breadboard.

Listing 1. Function to configure the SERCOM module as an SPI master.

```

void configure_spi_master(void)
{
    struct spi_config config_spi_master;
    struct spi_slave_inst_config slave_dev_config;
    spi_slave_inst_get_config_defaults(&slave_dev_config);
    slave_dev_config.ss_pin = PIN_PA05;
    spi_attach_slave(&slave, &slave_dev_config);
    spi_get_config_defaults(&config_spi_master);
    config_spi_master.mux_setting = EXT1_SPI_SERCOM_MUX_SETTING;
    config_spi_master.pinmux_pad0 = EXT1_SPI_SERCOM_PINMUX_PAD0;
    config_spi_master.pinmux_pad1 = PINMUX_UNUSED;
    config_spi_master.pinmux_pad2 = EXT1_SPI_SERCOM_PINMUX_PAD2;
    config_spi_master.pinmux_pad3 = EXT1_SPI_SERCOM_PINMUX_PAD3;
    config_spi_master.transfer_mode = SPI_TRANSFER_MODE_0;
    config_spi_master.data_order = SPI_DATA_ORDER_MSB;
    config_spi_master.mode_specific.master.baudrate = 250000;
    spi_init(&spi_master_instance, EXT1_SPI_MODULE, &config_spi_master);
    spi_enable(&spi_master_instance);
}

```

Listing 2. The rather lengthy callback function for the RTC in calendar mode.

```

void rtc_match0_callback(void)
{
    adc_start_conversion(&adc_instance);
    while(adc_read(&adc_instance, &data) == STATUS_BUSY){}
    volt = data * 0.000805;
    measure_result = 25 + (volt - 2.945) / 0.01;
    rtc_calendar_get_time(&rtc_instance, &time);
    spi_select_slave(&spi_master_instance, &slave, true);
    spi_write_buffer_wait(&spi_master_instance, &write_enable, 1);
    spi_select_slave(&spi_master_instance, &slave, false);
    delay_ms(10);
    spi_select_slave(&spi_master_instance, &slave, true);
    spi_write_buffer_wait(&spi_master_instance, &write_command, 1);
    spi_write_buffer_wait(&spi_master_instance, &address[i], 1);
    spi_write_buffer_wait(&spi_master_instance, &time.hour, 1);
    spi_write_buffer_wait(&spi_master_instance, &time.minute, 1);
    spi_write_buffer_wait(&spi_master_instance, &time.second, 1);
    spi_write_buffer_wait(&spi_master_instance, &measure_result, 1);
    spi_select_slave(&spi_master_instance, &slave, false);
    i++;
    alarm.mask = RTC_CALENDAR_ALARM_MASK_SEC;
    alarm.time.second += 10;
    alarm.time.second = alarm.time.second % 60;

    if(i < 10)
    {
        rtc_calendar_set_alarm(&rtc_instance, &alarm, RTC_CALENDAR_ALARM_0);
    }
    else
    {
        measure_ended = 1;
        port_pin_set_output_level(LED0_PIN, 0);
    }
}

```


then transferred to the SERCOM module using the command

```
spi_attach_slave(&slave, &slave_dev_config);
```

If there are several slaves this process needs to be repeated to assign an SS pin for each.

Next we turn to `config_spi_master`. Again we start by populating the structure with the default settings, and then set up the remaining pins to be used for the interface. SPI mode 0 is selected using the command

```
config_spi_master.transfer_mode = SPI_TRANSFER_MODE_0;
```

and then the bit order ('MSB first') is set:

```
config_spi_master.data_order = SPI_DATA_ORDER_MSB;
```

and finally the baud rate is configured to 250 kHz:

```
config_spi_master.mode_specific.master.baudrate = 250000;
```

Then we transfer the settings in the structure to the SERCOM0 module (which here goes under the name of `EXT1_SPI_MODULE`) and enable the peripheral.

The code also declares various variables and arrays. For example, the command bytes for the SPI EEPROM (see the **Text box**) are kept in an array, as are the ten addresses in the array address. More on this shortly.

Programs under development often change rapidly. For this reason we have in some cases used an array even when only one variable is being stored, as this makes it easy to expand to multiple variables at a later stage, which simplifies modifying the program.

The most important parts of the program, whose flowchart is shown in **Figure 5**, are the main function and the RTC counter ISR. First we set the control register of the 25LC040 so that we have access to the whole of its address space, and, of course, configure the other peripheral modules of the SAM D20. For test purposes we set the RTC's time to 13:45:34 on 12.10.2015. The microcontroller now begins to take temperature readings using the ADC at ten second intervals. It stores the time in hours (1), minutes (2) and seconds (3) along with the temperature (4) in exactly that order in four registers, which explains the steps of four in the addresses in the array address. The corresponding code appears in the RTC callback ISR (**Listing 2**) which is called whenever an alarm is triggered, starting at 13:45:35 and recurring every ten seconds thereafter. On each call the variable `i` is incremented until it reaches ten, which indicates that a sequence of readings is complete. The sequence takes a total of $10 \times 10 = 100$ seconds to run.

At the end of a sequence of readings the if-statement at the end of the ISR causes LED0 on the board to light to indicate that logging is complete. The alarm is not reset (and hence the ISR will not be called again) and the variable `measure_ended` is set to 1. This variable acts as a flag to the main loop that allows it to read the stored data from memory. The stored temperature values can now be read out from memory as often as desired and sent out over the UART interface, which in turn is connected to the host PC via the embedded debugger (EDBG).

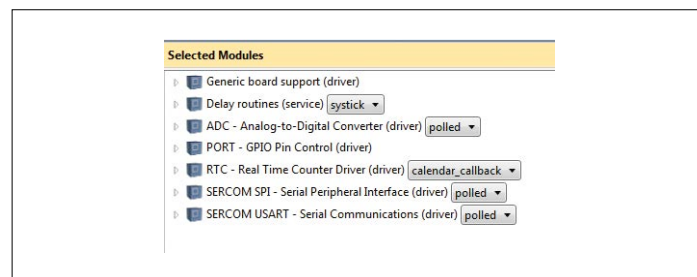


Figure 4. The libraries that need to be included in the project. In the ASF wizard the same library handles both SPI slave and SPI master modes.

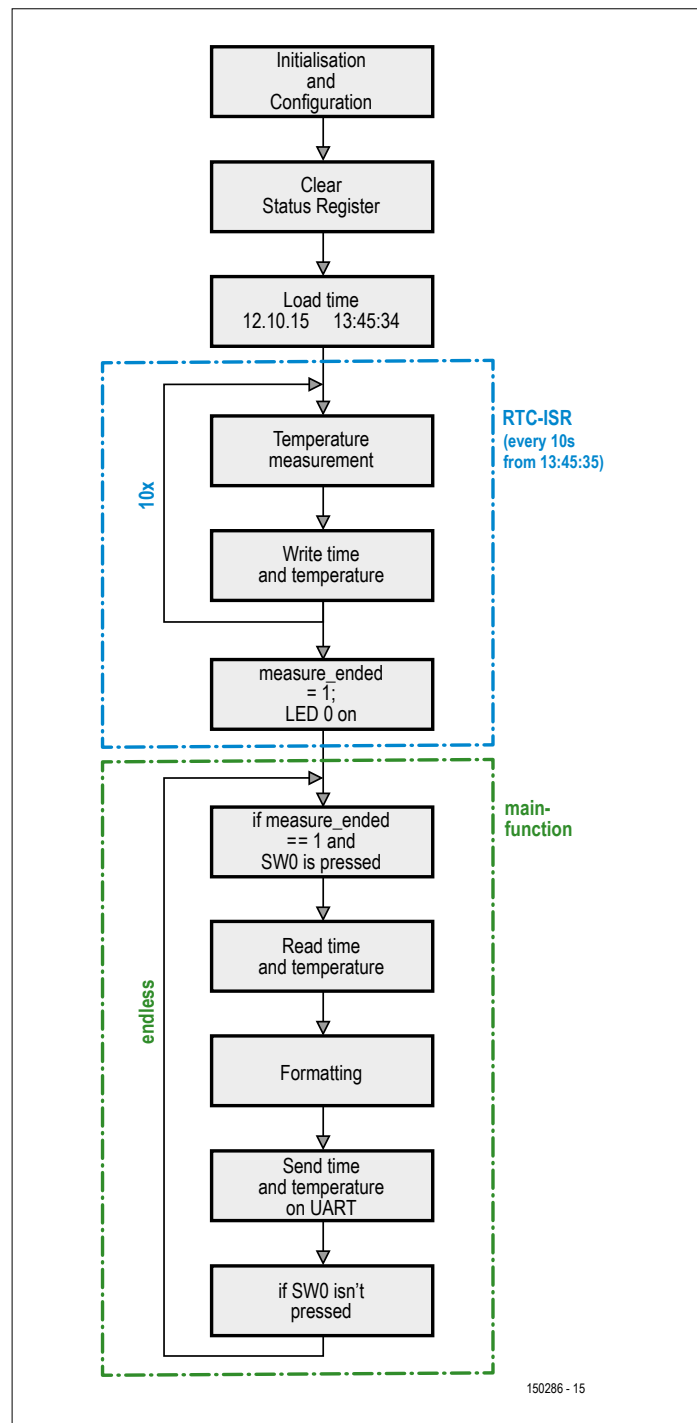


Figure 5. Program flowchart.

This function is implemented in the if-statement in the main loop which checks whether button SW0 on the board has been pressed: see **Listing 3**.

In the for-loop that follows the variable `i` is again incremented from 0 to 9. For each pass around the loop the stored data are read from the next four registers in the EEPROM and output over the UART interface. Before sending the raw values must be converted into a suitable printable format:

```
sprintf(usart_buffer_time, "Time: %d:%d:%d\n", read_data[0], read_data[1], read_data[2]);
```

```
sprintf(usart_buffer_temperature, "Temperature: %d degree\n", read_data[3]);
```

As is clear from this code, the data read from the EEPROM is stored in the array `read_data`.

After the for-loop there is a simple while-loop

```
while(!port_pin_get_input_level(BUTTON_0_PIN)){}
```

which waits for SW0 to be released before allowing the next transmission of stored data and times. This way it is not possible to cause the same data to be sent several times by accidentally holding the button down.

We have not yet looked at the SPI commands in detail, but they are easy to understand. There are really only two types of command in the program. The first,

```
spi_write_buffer_wait(&spi_master_instance, &address[i], 1);
```

transmits a buffer of arbitrary length over the SPI port. It requires a pointer to the SPI instance structure, a pointer to the buffer containing the data, and finally a count of the number of bytes to be sent. The second command,

```
spi_read_buffer_wait(&spi_master_instance, &read_data, 4, 0x00);
```

reads a buffer of arbitrary length over the SPI port. Again, it requires a pointer to the SPI instance structure, a pointer to the buffer to receive the data, and a count of the bytes to be received. Finally there is a 'dummy byte', which is transmitted while reception is in progress, but which serves no other purpose.

A delay of 10 ms should in general be inserted between these commands to allow time for the EEPROM IC to carry out its internal write operation.

The command

```
spi_select_slave(&spi_master_instance, &slave, x); // x = true/false
```

should be used before and after each data transfer over SPI to set the SS signal respectively low and high, as required by the SPI protocol.

This project, under the name of 'Temperature-Datalogger', can be downloaded from [2] to try it out. You will need to start a terminal emulator program on the PC, reset the microcontroller and then wait 100 seconds until LED0 lights to indicate that the sequence of readings is complete. Then press SW0 to read out the ten stored temperature values and their timestamps.

Room for improvement

It is hard to believe that the 215 lines of code that make up this project occupy only 7.2 % of the program memory of the microcontroller: welcome to the world of ARM!

Although this has been the largest project in the course so far, having tried it out you should not have had too much difficulty understanding it. We have looked at the SPI bus and learned how to drive an EEPROM. The ideas can be transferred to your own project, or you can refine this one: for example, the timing of readings could be changed, and the overall logging period could be increased. The temperature values could be stored

Listing 3. A look at the main loop.

```
if (measure_ended == 1 && port_pin_get_input_level(BUTTON_0_PIN) == 0)
{
for (i=0; i<10; i++)
{
spi_select_slave(&spi_master_instance, &slave, true);
spi_write_buffer_wait(&spi_master_instance, &read_command, 1);
spi_write_buffer_wait(&spi_master_instance, &address[i], 1);
spi_read_buffer_wait(&spi_master_instance, &read_data, 4, 0x00);
spi_select_slave(&spi_master_instance, &slave, false);
delay_ms(10);
sprintf(usart_buffer_time, "Time: %d:%d:%d\n", read_data[0], read_data[1], read_data[2]);
sprintf(usart_buffer_temperature, "Temperature: %d degree\n", read_data[3]);
usart_write_buffer_wait(&usart_instance, (uint8_t *)usart_buffer_time, 15);
usart_write_buffer_wait(&usart_instance, (uint8_t *)usart_buffer_temperature, 23);
}
}
while(!port_pin_get_input_level(BUTTON_0_PIN)){}
```


to higher precision, as could the date. Bear in mind that at most 16 bytes (one page) can be written to or read from the EEPROM in a single operation.

As in the last-but-one installment of this course, which dealt with the I²C bus, an accompanying project will appear in the Elektor Forums [4] showing how to SAM D20 can be used in SPI slave mode, with the help of an Arduino Uno.

In the next installment we will describe several smaller projects which demonstrate some of the libraries we have so far neglected to cover for reasons of space. There will also be a few tips and tricks for using the SAM D20 and Atmel Studio. Watch this space! ◀

(150286)

Web Links

- [1] www.atmel.com/images/Atmel-42129-SAM-D20_Datasheet.pdf
- [2] www.elektormagazine.com/150286
- [3] www.atmel.com/Images/Atmel-42115-SAM-D20-Serial-Peripheral-Interface-Driver-SERCOM-SPI_Application-Note_AT03255.pdf
- [4] <http://forum.elektor.com/viewtopic.php?f=2698569&t=2715200>
- [5] https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus
- [6] <http://ww1.microchip.com/downloads/en/DeviceDoc/22064D.pdf>

The 25LC040

This small and low-power device from the eight-pin 25LC family of EEPROMs offers a storage capacity of 4 Kbits (512 bytes) and is very easy to drive over SPI at clock rates of up to 2 MHz. EEPROM is a non-volatile storage technology and has a number of advantages, in particular a high write speed. In the case of this device, the maximum write time is 5 ms.

Figure 6 shows the internal structure of the EEPROM. There are two active-low inputs: /HOLD and /WP. The first of these can be used to pause an SPI transfer that is already underway, while the second prevents the device from being written to over SPI. If these functions are not used, the pins must be pulled high using pull-up resistors. The remaining pins are for power and for the SPI bus connections.

Write and read sequences and commands are very simple, as shown in Figure 7 and Table 1. First the appropriate command is sent, followed by the memory address. The read or write data then follow. Up to sixteen bytes can be sent or received in one transfer, which lasts until the

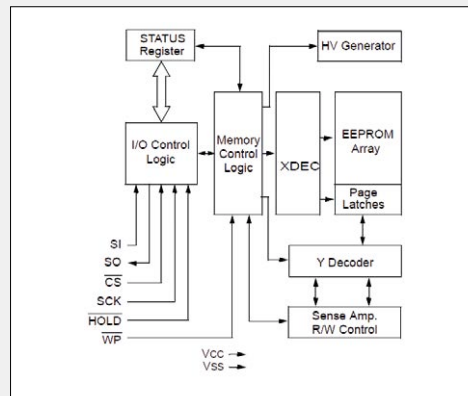


Figure 6. Internal structure of the 25LC040. (Source: Microchip)

master (the microcontroller) brings the SS pin (here /CS) back to a high level. Before each write operation the WREN command must be sent to enable writing; also, the protection bits in the status register must be appropriately set to allow access to the memory area in question.

The 25LC040 is compatible with the larger-capacity devices in the same family of EEPROMs, the only difference being that they require more address bytes [6].

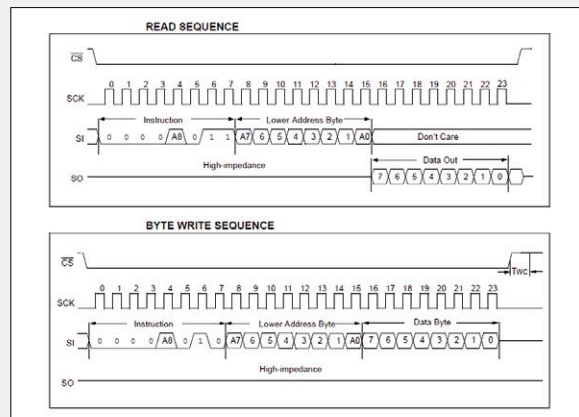


Figure 7. Read sequence (above) and write sequence (below). Although not shown here, up to sixteen bytes can be sent or received in one transfer. A8 is the ninth bit of the address and for our purposes is always low. (Source: Microchip)

Table 1. The main commands supported by the 25LC040 (simplified)

Command	Description
READ (0x03)	Read data from the device starting from the selected address.
WRITE (0x02)	Write data to the device starting from the selected address. The write enable latch must first be set (see WREN).
WRDI (0x04)	Disable the write enable latch, preventing write access to the EEPROM.
WREN (0x06)	Enable the write enable latch, allowing write access to the EEPROM.
RDSR (0x05)	Read from status register, including flags that indicate which memory regions are write-protected.
WRSR (0x01)	Write to status register, including flags that indicate which memory regions are write-protected. This command can change the write-protection settings.

DesignSpark Mechanical CAD Tips & Tricks (3)

Add a Bill of Materials

By Neil Gruending (Canada)

Learn how to add a Bill of Materials to a DesignSpark Mechanical 3D model.

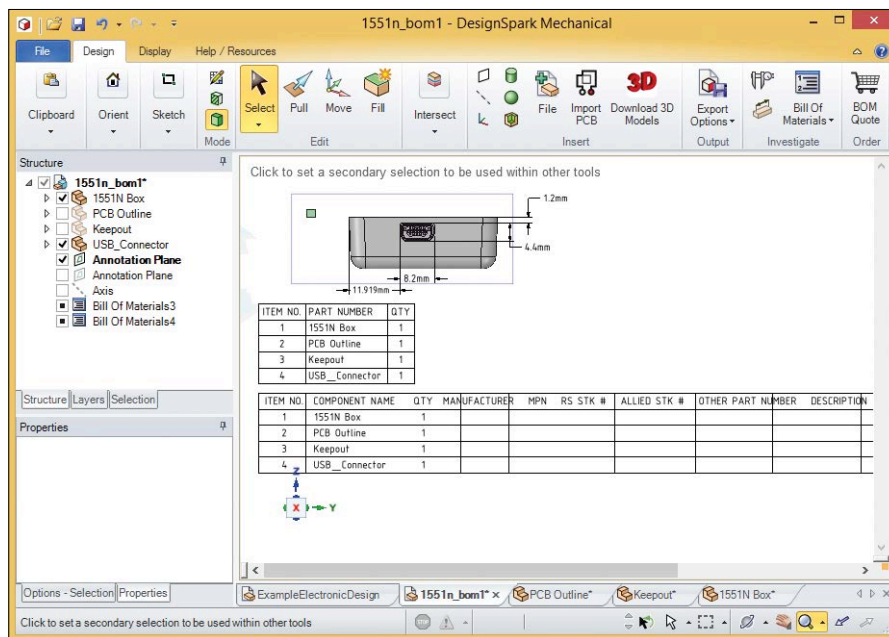


Figure 1. BOM examples.

Adding a Bill of Materials (BOM) to a DesignSpark Mechanical model is easy. Let's see how to add one to the modified PCB enclosure that we added dimensions to last time.

Adding the BOM

The first thing we need is a plane to put the BOM on because it's a two dimensional object. It can be any plane in a design but usually you would use an annotation plane. In our example we will use the annotation plane that we used for the dimensions but you could also make a new one using either the Dimension or the Plane tool if you wanted to. Once you know where you want to put it you can add a Full Detail or Top Level BOM using the Bill of Materials tool in the Investigate menu like in **Figure 1**.

The top table is the Top Level BOM. It shows all of the design components and their names in the Part Number column. The bottom table is a Full Detail BOM and is usually the more useful of the two. It shows all of the design components along

with manufacturer part information, RS Components stock numbers and Allied Electronics part numbers. As you can see we will have to add the manufacturing information to the components before the Full Detail BOM will be useful. But if you download components from RS then all of the manufacturing information is already part of the component.

Adding Manufacturing Information

The DesignSpark BOM tool uses custom property fields to store the manufacturing information for a component. Adding all of those fields every time you draw a component can be tedious so DesignSpark has the Edit Ordering Information tool in **Figure 2** to add them for you. All you have to do is right-click on the component and choose Edit Ordering Information from the menu.

Once you've added all of the information, click on Save and the BOM fields will be added to the component properties. **Figure 3** shows our example after I added the BOM fields for

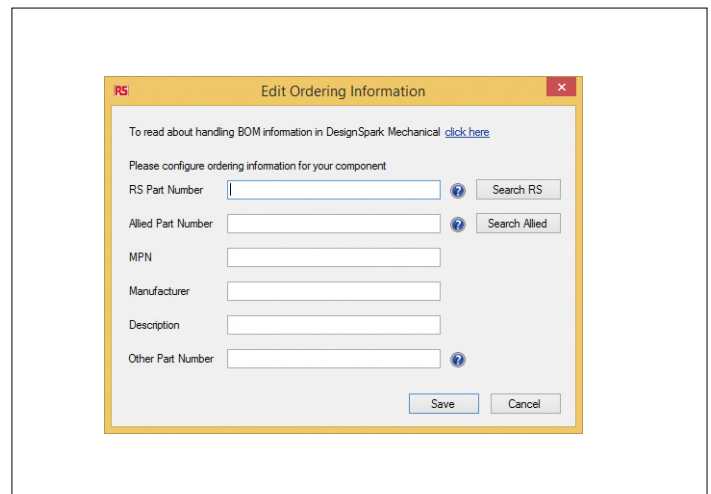


Figure 2. Edit Ordering Information window.

in collaboration with **DESIGNSPARK**

the Hammond 1551 enclosure component. The BOM table was automatically updated and in the lower left corner you can see the fields that were added to the component. If you ever need to edit the manufacturing information, you can either use the Edit Manufacturing Information tool again or directly edit the fields in the component properties window.

You can also control if a component is visible in the BOM or not. This is helpful in our example because we can hide the keepout component from the BOM even though it's part of the model for various reasons. First you right click on the component and then choose Add to Bill of Materials to add the @BomInclude property to the component. Now change it from True to False and it will be hidden from the BOM like in **Figure 4**.

Export to CSV

Once you've finished with setting up the BOM it's easy to export it to a Comma Separated Value (CSV) format so that you can open it a spreadsheet application like Microsoft Excel. First you click on a BOM table to select it and then DesignSpark Mechanical will draw a box around it with dashed lines. Now you can choose the CSV export option in the Bill of Materials menu and DesignSpark will export a BOM as a .csv file in the project directory with same name as the .rdoc design file. It will also automatically open the default external program associated with CSV files where you can edit the file as needed.


Online BOM Quotation

DesignSpark Mechanical also has an online quotation tool that will automatically upload your BOM to RS Components or Allied Electronics so that you can easily order the parts for your design. All you have to do is choose BOM Quote from the Order menu. The quotation tool will then generate its own BOM to upload so you don't have to select a BOM table before running the tool. **Figure 5** shows the uploaded BOM for our example.

The website, Allied Electronics in my case since I'm in Canada (www.alliedelec.com; Allied is the US/Canadian/Mexican branch of RS Components, Ed.), will then try to match all of the part numbers in your design to the available parts. In this case it found the Hammond enclosure where I had entered the proper RS and Allied stock numbers. But you don't always have to have the stock numbers in advance. For example, I only entered the manufacturer (Molex) and the part number prefix (54819) for the USB connector. DesignSpark will warn you when you don't enter a RS Components or Allied stock number but it's okay to ignore it. The Allied site then gave me a list of possible matches for the Molex part number so that I could choose the complete part number.

Conclusion

DesignSpark Mechanical has some really easy to use BOM tools to help speed up your design process. I used a pretty simple example here but the real value of the BOM tools is when you have larger, more complex designs.

Give it a try! 

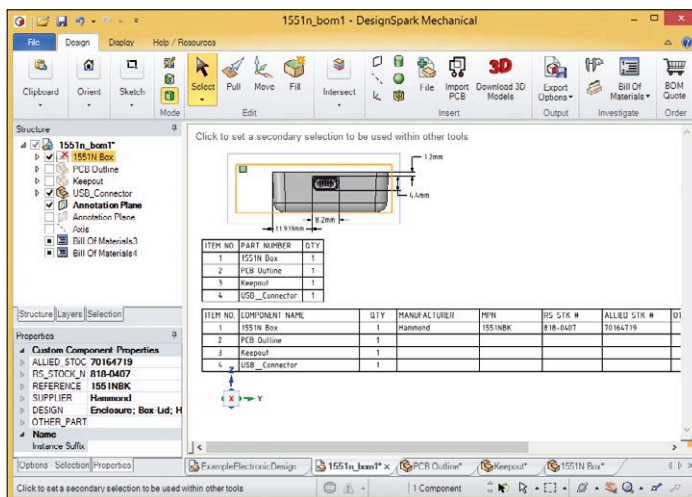


Figure 3. Updated enclosure information.

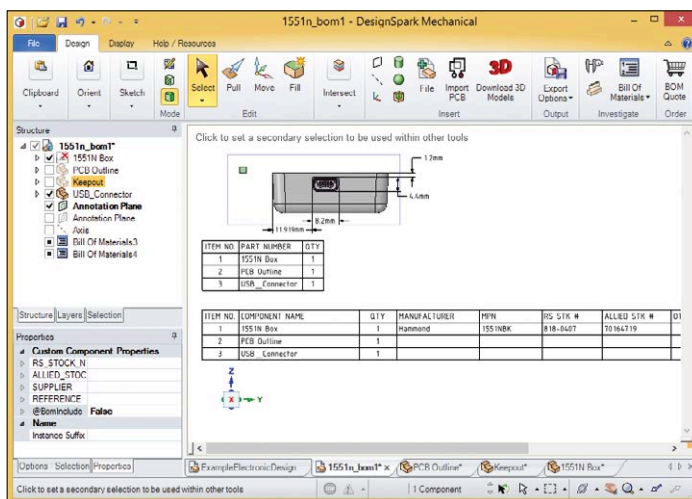


Figure 4. BOM examples.

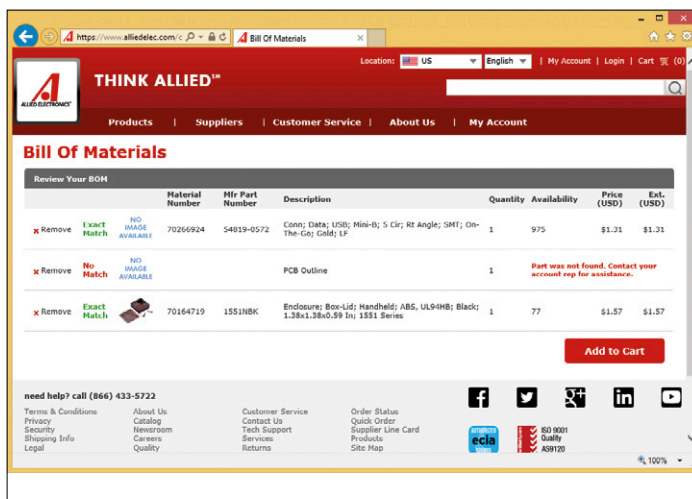


Figure 5. Online quotation tool.

(150294)

PIC[®] Assembler Crash Course

(3) PIC replacement for 555 timer

By **Miroslav Cina** (Germany) miroslav.cina@t-online.de

The first part of this series introduced the basics of programming in Assembler and the hardware fundamentals of the PIC controller used. Part two dealt with electronic dice. In this final part we discover how we can get a modern microcontroller to simulate a standard IC such as the well-known 555 timer.

There's no need to waste words describing the NE555 timer. This IC has been produced since 1972 and up to now has held the record for the quantity of units made, even though it's not even 'digital'. Its remarkable versatility means that it is by no means confined to its intended applications as a squarewave generator or monostable.

The standard bipolar version generates signals up to 500 kHz, whilst modern successor chips, like the LMC555 from TI, can produce frequencies right up to 3 MHz. But squarewave signals and suchlike can also be produced using microcontrollers as modern-day 'do everything' ICs. We shall now explain how to use a PIC12F675 as a 555 replacement. This time round the focus is on its function as a squarewave oscillator. Other functions such as monostables and so on can of course be achieved in software too if you feel like being creative.

1. Minimalist oscillator

For our first program we need just a microcontroller plus a couple of lines of code — and that's it. The code lets us set the frequency of the squarewave signal generated, from extremely

low right up to around 166 kHz. At lowish frequencies a microcontroller is simply outstanding, as it can generate stable periods or pulses with durations of one day, one month or even longer. The maximum frequency depends on the clock speed of the controller. As **Figure 1** illustrates, the circuitry could hardly be simpler.

The necessary program concerns itself only with controlling the output. For a squarewave signal with a mark-space ratio of 50 % we achieve the highest frequency with the following Assembler code:

```
output_1 bsf  GPIO,D'001' ;1us @ 4 MHz
          nop                ;1us @ 4 MHz
          nop                ;1us @ 4 MHz
          bcf  GPIO,D'001' ;1us @ 4 MHz
          goto output_1     ;2us @ 4 MHz
```

The first command sets output GP1 to logical '1'. In the commentary to this the run-time of the command is declared as 1 μ s at a clock rate of 4 MHz. After this we wait 2 μ s during two nop commands until the output is set back to logical '0'.

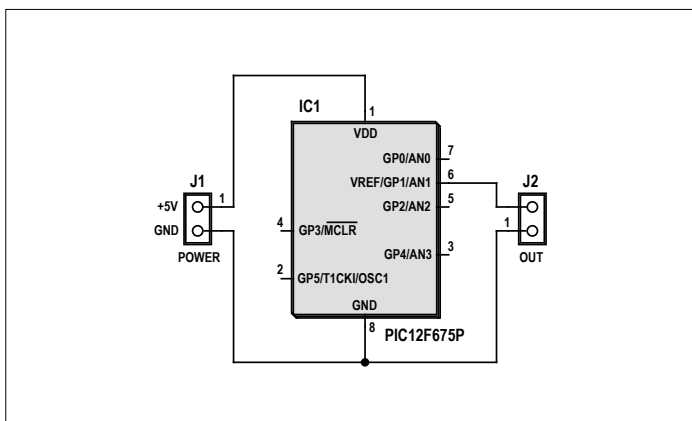


Figure 1. In the simplest version of a 555 replacement the PIC manages without any external connections.

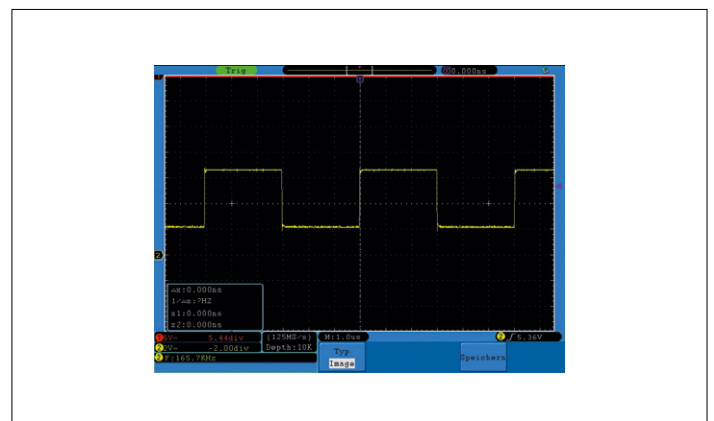


Figure 2. Oscillogram of the signal in example #1: clean and tidy 166 kHz with a duty cycle of 50 %.

The delay is necessary for reasons of symmetry, as reverting to the start of the loop using `goto` takes 2 μ s. With the internal oscillator clocked at 4 MHz, this endless loop produces an output frequency of around 166 kHz.

The oscillogram in **Figure 2** demonstrates the high quality of the squarewave signal produced in this way.

For lower frequencies we simply build in longer delays. To do this we have two possibilities: for the kHz region we add the appropriate number of additional NOPs. For 'slower' signals it's better to create an explicit delay loop that is initiated from the main loop. For a duty cycle of 0.5 the delays must be symmetrical. The following example demonstrates how we invoke the routine `wait_1` twice:

```
output_1 bsf  GPIO,D'001' ;1us @ 4 MHz
         nop                ;1us @ 4 MHz
         nop                ;1us @ 4 MHz
         call wait_1        ;predefined delay
         bcf  GPIO,D'001' ;1us @ 4 MHz
         call wait_1        ;predefined delay
         goto output_1     ;2us @ 4 MHz
```

Invoking this using a `call` command takes 4 μ s altogether (`call` itself takes 2 μ s and the reversion that follows in the routine takes a further 2 μ s). In the routine `wait_1` we determine the delay period required.

If an asymmetric duty cycle is called for, we build in asymmetric delays. The following code achieves this with `nop` commands:

```
output_1 bsf  GPIO,D'001' ;1us @ 4 MHz
         nop                ;1us @ 4 MHz
         nop                ;1us @ 4 MHz
;additional delay - additional 4  $\mu$ s -----
         nop
         nop
         nop
         nop
;-----
         bcf  GPIO,D'001' ;1us @ 4 MHz
         goto output_1     ;2us @ 4 MHz
```

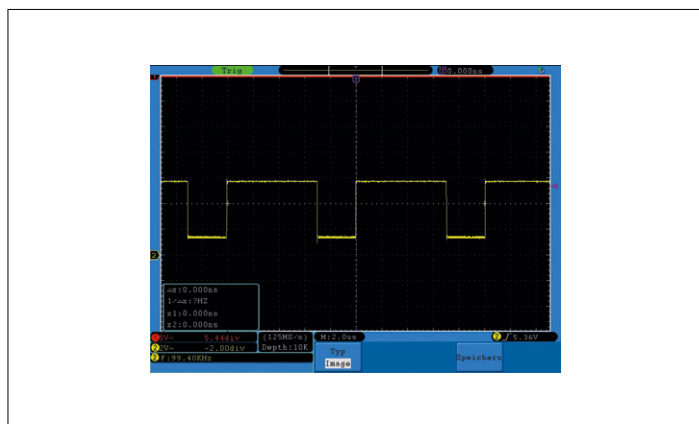


Figure 3. With four additional `nop` commands the frequency drops to 100 kHz and the 70 % duty cycle is no longer symmetrical.

The extra 4 μ s reduce the frequency to around 100 kHz. Since the 1-phase now requires 7 μ s and the 0-phase only 3 μ s, however, the duty cycle of 70 % seen in **Figure 3** arises.

Some more about timing: normally all commands require four clock pulses. The execution time is therefore very easy to calculate as $t = 4 / \text{clock frequency}$.

Exceptions arise, for example, with the `goto`, `call` or `return` commands that manipulate the program counter, as these require eight clock pulses and consequently double the execution time.

It's worth noting that you can adjust the accuracy of the internal clock generator, factory-calibrated to ± 1 %. This calibration value laid down in the chip can be altered when necessary. We initiate this with the following code:

```
bsf  STATUS,RP0
call H'3FF'
movwf OSCCAL
bcf  STATUS,RP0
```

It goes without saying that we can also write our own value into the Register `OSCCAL` and thereby alter the actual clock. Now that the principle is clear, check out **Listing 1** for the complete code to transform a PIC12F675 into a 166 kHz square-wave generator with a duty cycle of 0.5.

First the controller in use is established and then configured from `_U002`. In the process, the internal oscillator is activated using the three LSBs with the value '100'. It is established that `MCLR` is acting as an input and not, say, as a reset pin. The next four commands (from `bsf` to `clrf`) deactivate the unnecessary analog functions and set Pin GP1 as output. The following three commands `call` to `bcf` calibrate the internal oscillator as already described. The closing main loop from `bsf` to `goto` consists of only five commands.

2. Setting the frequency

The schematic in **Figure 4** includes an additional pot or trimpot (potentiometer) P1, with which we can set the output frequency from approx. 885 kHz to 1.45 MHz. However, the way

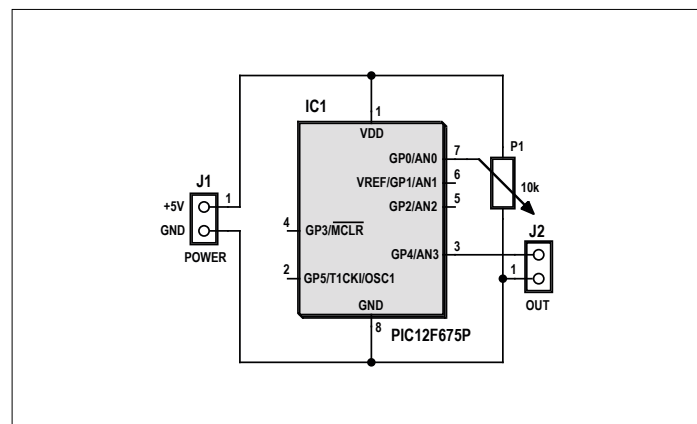


Figure 4. Adding a trimpot makes the frequency adjustable.

Table 1. ADCON0 Register

Denotes	ADFM	VCFG	-	-	CHS1	CHS0	GO/DONE	ADON
Bit	7	6	5	4	3	2	1	0

we produce the signal is totally different from that in the first example. Here we demonstrate the use of the *CLKOUT* feature together with how we handle the integrated A/D converter.

Previously squarewave signals were produced by alternately setting and resetting the output pin. Now we're going to let the hardware do this for us. The PIC12F675 offers the facility to direct the clock frequency (divided by a factor of 4) direct to an output. If we now set the three LSBs of the Configuration Register to '101' instead of '100', Pin GP4 can be used for *CLKOUT*. With a clock rate of 4 MHz we get a signal of 1 MHz and a duty cycle of 50 %.

To alter the frequency, the internal oscillator is recalibrated (using other values for the *OSCCAL* Register), as mentioned previously. The value of the 8-bit Register ranges between 00h and FFh. The internal oscillator can typically be pulled from frequencies between 3.0 and 5.8 MHz. A quarter of this produces the output frequency range already mentioned. Some additional code is required to enable the microcontroller to recognize the position of the pot. Three Registers are relevant to this:

- **ADCON0**: Configuration of the ADC.
- **ADRESH, ADRESL**: MSB/LSB of the 16-bit ADC results.

- **ANSEL**: Analog Select Register; this determines the sample rate and which Pin becomes an analog input.

The significance of the Bits in the *ADCON0* Register can be checked out in **Table 1**. Bit 7 is the so-called A/D Result Formed Select Bit (ADFM). The integral ADC has a resolution of 10 Bits but the result has a width of two 8-bit Registers = 16 bits, of which only 10 Bits are significant. The remaining 6 Bits have the value '0'.

The value of ADFM decides the position of the overflow Bits. When ADFM = 0 the result is aligned left, i.e. the six LSBs of the *ADRESL* Register are set to '0'. Consequently the eight MSBs of the result are deposited in the *ADRESH* Register. On the other hand, the two remaining LSBs go in the two MSBs of *ADRESL*. If ADFM = 1 it's the other way round and the result is now aligned right.

VCFG selects the reference voltage of the ADC. If VCFG = 1, then the V_{ref} Pin (GP1) is used. When VCFG = 0 it's simpler and the supply voltage V_{DD} is used as reference.

The two Bits CHS1 and CHS0 select the Pin that acts as input for the ADC. If both are '0', channel AN0 and Pin GP0 are selected. Go/Done initiates an A/D conversion when we set it to '1'. We can check afterwards whether the conversion is already complete. Then this bit has the value '0'.

Listing 1.

```

;*****
;*      NE555 Replacement - Example 1      *
;*      v 1.00 - 19.06.2015                *
;*****
;*      Microcontroller: PIC12F675        *
;*      Oscillator: internal                *
;*****
;Pin connections:
;GP0 --> N/C
;GP1 --> Output
;GP2 --> N/C
;GP3 --> N/C
;GP4 --> N/C
;GP5 --> N/C
;-----
;Chip configuration:
INCLUDE "P12F675.INC"
_U002 EQU B'00000110000100'
        ;MCLR = input/internal osc.
__CONFIG U002
;-----
;analog functions off, GP1 = output
        bsf STATUS,RP0 ;switch to regist. bank 1
        movlw B'11111101'
        ;GP1 = output; GP0; GP2 - GP5 = input
        movwf TRISIO
        cllf ANSEL ;GPIO are digital I/O's
;-----
;oscillator calibration
        call H'3FF'
        movwf OSCCAL
        bcf STATUS,RP0 ;switch back to Bank 0
;-----
;Main Loop
;-----
output_1 bsf GPIO,D'001' ;1us @ 4 MHz
        nop ;1us @ 4 MHz
        nop ;1us @ 4 MHz
;additional delay - additional 4 us -----
        nop
        nop
        nop
        nop
;-----
        bcf GPIO,D'001' ;1us @ 4 MHz
        goto output_1 ;2us @ 4 MHz
;-----
END

```

Listing 2.

```

;*****
;*      NE555 Replacement - Example 2 with P1      *
;*      v 1.00 - 19.06.2015                      *
;*****
;*      Microcontroller: PIC12F675              *
;*      Oscillator: internal                    *
;*****

;Pin connections:
;GP0 --> Poti
;GP1 --> N/C
;GP2 --> N/C
;GP3 --> N/C
;GP4 --> CLKOUT
;GP5 --> N/C
;-----
;Chip configuration:
INCLUDE "P12F675.INC"

_U005      EQU  B'00000110000101'
           ;MCLR = input / internal osc. + CLKOUT
__CONFIG  _U005
;-----
;setup ADC -----
;Result left adjusted, VDD = reference, GP0 = input,
activate ADC
           movlw  B'00000001'
; ADFM      = 0 --> result is left justified
; VCFG      = 0 --> Vdd = voltage reference
; CHS1:CHS0 = 00 --> Channel 00 (GP0)
; GO        = 0 --> no start now

```

```

; ADON      = 1 --> switch the module ON
           movwf  ADCON0
;Samplerate, GP0 = analog input
; movlwB'00010001'
; ADCS<2:0> = 001 --> ADC clock = FOSC/8
; ANS3 = 0
; ANS2 = 0
; ANS1 = 0
; ANS0 = 1 --> analog input on AN0 (GP0)

           bsf   STATUS,RP0
           movwf ANSEL
           bcf   STATUS,RP0

;-----
;Main Loop
;-----
;start A/D conversion -----
main_loop bsf  ADCON0,D'001'
w_adc     btfsc ADCON0,D'001'
           goto  w_adc

;modify c'clock -----
           movf  ADRESH,0
           bsf  STATUS,RP0
           movwf OSCCAL
           bcf  STATUS,RP0
           nop
           nop
           goto main_loop

;-----
END

```

Finally we use *ADON* = 0 to shut down the complete A/D section (to save current) and *ADON* = 1 to reactivate it. The *ADRESH* and *ADRESL* Registers contain the ADC result, as already mentioned.

The significance of *ANSEL* is given in **Listing 2**.

The first variation from Listing 1 lies in the value of '101' for the three LSBs of the configuration Register. Consequently at Pin GP4 we have a quarter of the clock frequency at our disposal. In the following section we configure the ADC. First the result is aligned to the left. If we use only *ADRESH* and ignore *ADRESL*, we'll get a ready formatted 8-bit result. In addition V_{DD} is used as reference and GP0 as measurement input.

The next step will see the ADC clock frequency reset to one-eighth of the main clock and GP0 set as analog input. With that the Initialization process is over.

In the main loop we initiate an A/D conversion and then read in the result. Following this, we make the frequency of the internal oscillator dependent on the ADC result and set the position of the trimpot wiper with a resolution of 256 steps.

3. Interrupts and TIMERO

If we wish to generate a frequency that is both fixed and crystal-stabilized, we need to take a different (and more involved)

approach. There is no getting around the use of Interrupts and setting a Timer. Additional components in the schematic shown in **Figure 5** are the crystal itself plus its two loading capacitors. The software has two tasks to fulfill. On one hand, as previously, it needs to read in the setting of the trimpot and on the other, generate a precise squarewave signal that will not be

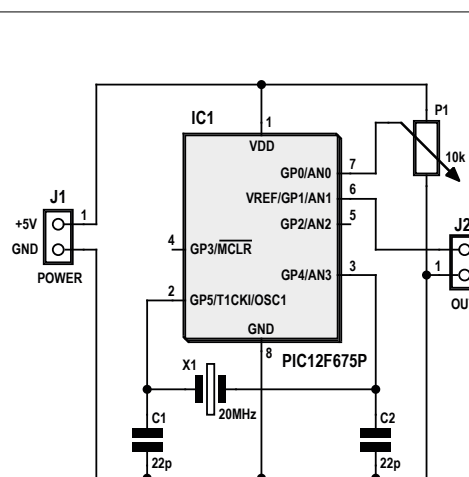


Figure 5. A crystal gives the squarewave generator greater stability.

Table 2. INTCON Register

Denotes	GIE	PEIE	T01E	INTE	GPIE	TOIF	INTF	GPIF
Bit	7	6	5	4	3	2	1	0

affected by frequency changes and suchlike. For this reason the code consists of two parts. We already know how to evaluate a pot. Producing the output signal employs two new functions of the microcontroller: Timer0 and Interrupts.

Timer0 can be used either as a timer or as a counter. In timer mode the associated Register is incremented periodically, being driven by clock pulses. As a counter it counts external events. For substituting the 555 chip we need to use the timer mode. The timer then generates the actual output signal. Here's how: At the outset a value is written into the *TIMER* Register TMR0. From then on this value is incremented continually until the value FFh is reached. Because we are dealing with an 8-bit Register, an overflow occurs with the next increment and the Register then contains the value 00h. The overflow is an event that triggers an Interrupt. In the interrupt routine we can generate the actual signal.

The interaction with Timer0 and its *TMR0* Register is totally uncomplicated. If we set the Bit TOCS to '0', Timer0 is used as

a Timer. From then onwards TMR0 is incremented with every Instruction cycle (clock rate divided by 4).

A couple of words more about Interrupts: all significant characteristics are controlled by the *INTCON* Register. The significance of its Bits can be inspected in **Table 2**. For our purposes only three of the Bits are important:

- **GIE**: Global Interrupt Enable. GIE = 1 activates and GIE = 0 deactivates the Interrupts.
- **TOIE**: Timer0 Interrupt Enable. Only when TOIE = 1 does an overflow of TMR0 produce an Interrupt.
- **TOIF**: Timer0 Overflow Indicator. An overflow of TMR0 makes the value of TOIF become 1. We can interrogate (poll) the Bit and it must then be reset to '0' in the code.

It's also vital that the Interrupt Routine begins always at memory location 004h. Two new Commands are relevant for Interrupts:

Listing 3.

```

;*****
;*   NE555 Replacement - Example 3 with Xtal   *
;*           v 3.07 - 05.07.2015             *
;*****
;*           Microcontroller: PIC12F675       *
;*           Oscillator: internal             *
;*****
;GP0 --> Poti
;GP1 --> Output
;GP2 --> N/C
;GP3 --> N/C
;GP4 --> Xtal
;GP5 --> Xtal
;-----
;Chip configuration:
INCLUDE "P12F675.INC"
_U003      EQU  B'00000110000010'
           ;MCLR = external xtal up to 20 MHz
__CONFIG U003
;=====
;Variable definitions
w_save0    EQU  H'20'
w_save1    EQU  H'A0'
stat_save  EQU  H'21'
v_delay    EQU  H'22'
v_value    EQU  H'23'
v_tmp      EQU  H'24'
;=====
;Set adress of subroutines
           ORG  H'000'
           goto rst_main
           ORG  H'004'
           goto int_main
;-----
;-----
;ISR
;Save status register -----
int_main movwf w_save0 ;save content of W-reg.
           ;could be Bank0 or Bank1
           swapf STATUS,0 ;move content of Status to W
           bcf STATUS,RP0 ;switch to Bank0
           movwf stat_save
           ;save original content of Status

;GPIO toggle -----
           movf v_value,0
           movwf GPIO
           incf v_value,1

;Store ADC value in TMR0 -----
           movf v_delay,0
           movwf TMR0
           bcf INTCON,T0IF

;Restore status register -----
           swapf stat_save,0 ;restore old Status to W
           movwf STATUS
           ;push content to Status register
           ;select old Bank
           swapf w_save0,1 ;rebuild content of W-reg.
           swapf w_save0,0 ;restore W register
           retfie
;-----
;ISR
rst_main
...
;missing code
...
;-----
END

```

SWAPF

This command swaps (exchanges) the two Nibbles of a Register. For a value of E4h, this changes after the swap to 4Eh. For Interrupts it is also important that the *STATUS* Register remains untouched by the execution of this command. The syntax is:

```
swapf f,d
```

The Register address 'f' can return values from 00h up to 7Fh, and 'd' can be either '0' or '1'. If d = 0, the result is saved in the *W* Register. If d = 1, then the result is written to the Register instead.

RETFIE

RETFIE is an acronym of **RE**TURN **FR**oM **IN**TERRUPT. The Interrupt Service routine is ended normally by this command and the main program is continued. No parameters are passed along with RETFIE (or with RETURN). The syntax is:

```
retfie
```

When an Interrupt is initialized, Bit GIE of the *INTCON* Register is set automatically to '0' and other Interrupts are disabled. The Command RETFIE resets Bit GIE back to '1'.

Now to the code in **Listing 3**: the first thing that strikes us is that CLKOUT is not used here. In addition the three LSBs of the *CONFIGURATION* Register are set to '010', enabling the use of higher frequency crystals.

Next follows a section where Variables are declared. Here we define the central Variables *v_delay*, *v_value* and *v_tmp* as well as *w_save0* and *w_save1* plus *stat_save* used for processing Interrupts.

The next section lays down the start addresses of two Sub-routines together with the ORG Directives. You haven't come across these Directives yet but their function is simple to explain: ORG sets the address of the following command on the value of a parameter. In our case *rst_main* is nailed to address 000h and *int_main* to 004h. As the Command goto occupies only one Byte, however, the Addresses 001h to 003h remain unused. Therefore we deploy the Directive, because the Vectors (target addresses) for Reset and Interrupt

Table 3. Output frequencies using crystal frequency as the parameter

Xtal frequency	f _{out} min.	f _{out} max.
2 MHz	237 Hz	4.63 kHz
4 MHz	474 Hz	9.26 kHz
20 MHz	2,370 Hz	46.30 kHz

must always be 000h and 004h.

Before we get to the ISR (Interrupt Service Routine) *int_main*, we should clarify exactly what an Interrupt is. In principle this happens when a defined event occurs to interrupt the execution of a program, and an ISR is executed instead. Following completion of this, we revert to the previous location in the main program and continue. This is similar to invoking a sub-program, except that this can take place any location desired in the program. In the process it is entirely possible that the program has just executed an arithmetic operation that has set the individual Bits of the *STATUS* Register, yet unfortunately an Interrupt has occurred before the BTFS Command following. As the ISR must not interfere with the course of the main program, it must save the contents of the *STATUS* Register before its own task and finally restore them (before returning to the program). Consequently the ISR turns out to be more comprehensive than we might have imagined.

It becomes apparent here that the Command SWAPF is important to have the ability to save the contents of a memory location in the *W* Register. At the start of the ISR the *W* and the *STATUS* Registers are saved. The first Command movwf *w_save0* saves the *W* Register to memory location *w_save0*. As we don't know whether the main program just operated with Bank 0 or Bank 1 at the start of the Interrupt, it is important to keep clear both *w_save0* (in this case 20h) as well as *w_save1* (A0h here). The Command movwf leaves the content of the *STATUS* Register unaltered by the way. The next three Commands save the contents of the *STATUS* Register to the memory location provided for this — *stat_save* (21h) in Bank 0. In this way we save everything that needs to be restored subsequently.

Now comes the actual work of the ISR. With each Interrupt the Variable *v_value* is output to the GPIO Port. The Variable

Listing 4.

```

;*****
;*      NE555 Replacement - Example 4 w/o code      *
;*          v 1.00 - 19.06.2015                    *
;*****
;*      Microcontroller: PIC12F675                 *
;*      Oscillator: internal, RC mode with external R *
;*****

;Pin connections:
;GP0 --> N/C
;GP1 --> N/C
;GP2 --> N/C

;GP3 --> N/C
;GP4 --> Output
;GP5 --> Poti
;-----
;Chip configuration:
INCLUDE "P12F675.INC"

_U004 EQU B'00000110000111'
;MCLR = input, RC mode, GP4 = CLKOUT
__CONFIG _U004
;-----
END

```

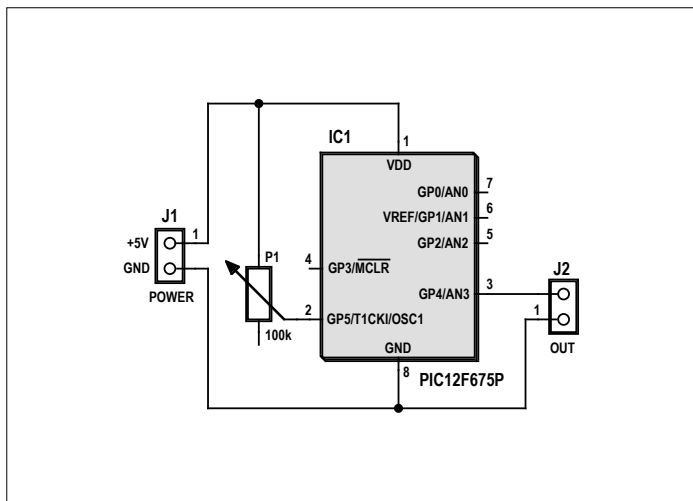


Figure 6. In the code-free version the PIC needs just one trimpot to set the duty cycle.

is then incremented. As Pin GP0 serves as a signal output (the level of which is set by the lowest-value Bit of the Variable), a change in level occurs with each Interrupt. Now only the Variable *v_delay* is copied into the *TMR0* Register. As the value of this Variable corresponds to the wiper position of the pot, the frequency output is determined in this way.

It only remains now to restore the original values of the *W* and *STATUS* Registers, after which the ISR can be exited using the Command *retfie*.

The frequency ranges covered with this circuit and code are dependent on the crystal frequency and can be seen in **Table 3**. A discussion of the missing part of the listing would be too lengthy for this article. But it goes without saying that the entire sample code, complete with the other listings, can be downloaded free from the web page for this article [4] and then inspected at your leisure.

4. PIC without software

A microcontroller without software — is this capable of doing anything at all? In fact, although the only additional component in **Figure 6** is a potentiometer, you can still manage to set a frequency of about 170 kHz to 3.5 MHz without using any bits and bytes.

If accuracy is not desperately important we can drive the internal oscillator in 'RC mode', without using a crystal. As in example #2, GP4 is used as a CLKOUT output. A resistor at

GP5 then determines the frequency. Since the oscillator frequency depends directly on the resistance, no code is required. The smaller the resistance at GP5, the higher the frequency. That said, if this resistance falls below roughly 3 kΩ, the oscillator stops. Shortly before this happens, the maximum frequency is around 3.5 MHz. At P1's maximum resistance (100 kΩ) the frequency is around 170 kHz. We can take this still further; with a resistance of 10 MΩ the frequency falls to approximately 2.1 kHz.

To be entirely correct, it does take one single word of data for the correct configuration of the chip and therefore something code-like is necessary. The associated listing 4 is decidedly short. Nothing is done apart from configuration. An excerpt from the corresponding .LST file proves that the length of the program is exactly '0':

MEMORY USAGE MAP ('X' = Used, '-' = Unused)

```
2000 : -----X----- -----
-----
```

All other memory blocks unused.

```
Program Memory Words Used:    0
Program Memory Words Free: 1024
```

The .LST file is created by running the source code through the assembler. It contains the code itself along with information from the assembler (for example how much program memory is occupied by the code).

Final thoughts

With that example using no code our Assembler Crash course has reached both its climax and its conclusion. I do hope you enjoyed it! Perhaps you will now get into PICKit3 or PICKit2 and try improving or extending one or two of the examples there. If you have any questions about this my e-mail address is at your disposal: miroslav.cina@t-online.de. ◀

(150393)

Web Links

- [1] PIC12F675: www.microchip.com/wwwproducts/Devices.aspx?product=PIC12F675
- [2] First part of this course: www.elektormagazine.com/130483
- [3] Second part of this course: www.elektormagazine.com/150274
- [4] Third part of this course: www.elektormagazine.com/150393

Coax Connectors

Peculiar Parts, the series

By **Neil Gruending** (Canada)

It's not accidental that coax (coaxial; co-axial) connectors are ideal for RF applications: after all, they surround the signal pin with a ground to control the connector impedance which is critical for RF. Over the years there's been a wide variety of connector styles so let's take a look at some of the classic types and their uses. Unexpectedly, some of the most common coax connectors that people are familiar with are the ones that were used for years in audio visual (A/V) applications. Phono or RCA plugs (**Figure 1**) have about a 10-MHz bandwidth which is more than enough to transmit analog audio and video signals in home theaters.

The threaded F connectors that are used for cable TV and Internet (**Figure 2**) are much higher bandwidth, up to 1 GHz in some cases, which is perfect for those higher frequency signals like satellite TV from the LNB. But there's more to coaxial connectors than your home theater.

One of the first coax connectors was designed in the 1930's and is typically called UHF or (Amphenol) SO-239 type. They are quite rugged and capable of handling relatively high power levels which makes them ideal antenna connectors for amateur and citizen's band (CB) radios. The upper frequency limit for UHF connectors is around 200 MHz though because of their nonlinear impedance at higher frequencies. With hindsight the 'U' in the type number is erroneous, since UHF is generally taken as starting at 300 MHz.

For higher frequencies we have BNC (Bayonet Neill-Concelman) connectors (**Figure 3**) which feature two bayonets and lock together with a quarter turn. Originally designed for the military, BNC connectors are now widely used in many applications with a bandwidth up to 2 GHz. One application that we're all familiar with is their use on test equipment like signal generators and oscilloscopes.

There's also a threaded version of BNC connectors called TNC (Threaded Neill-Concelman). The threads avoid the locking slots that limit the frequency range of BNC connectors so that a TNC upper frequency limit is about 11 GHz. This type of connector is also commonly used as an RF antenna connector. Both BNC and TNC connectors come in 50- Ω and 75- Ω impedances so it's important to use the correct connector for your application since they have slightly different connection dimensions. Normally you can tell them apart, though 75- Ω connectors have less dielectric than 50- Ω ones but this isn't guaranteed.

A more common type of threaded BNC variant is the N connector (**Figure 4**) which was invented by Paul Neill at Bell Labs in 1940. An N connector is substantially larger than a BNC but it is also capable of handling much higher power levels. It also comes in 50- Ω and 75- Ω versions but you can't mix them because a 50- Ω plug will damage a 75- Ω receptacle when they are connected together.

Another common connector is the SMA connector (**Figure 5**) which was invented in the 1960's. It's a lot smaller than a BNC which helps to extend its frequency range to about 18 GHz. It isn't rated for any significant power, or very many mating cycles but that's more than adequate for fixed installations.

This has just been a small sample of the many different types of coax connectors that are available. The classic coax connectors discussed here are still widely used today which is a testimony to their design and durability. ◀

(150341)



Please contribute your Peculiar Parts article, email neil@gruending.net



1



2



3



4



5

An Analog Robot

From concept to construction

By Tino Werner (Germany)

The commercial robot kit 'Tibo' packs lots of fun for young and old alike. A standard opamp lies at the heart of the design; you won't find a single line of code or a microcontroller anywhere. Pluggable resistors and capacitors allow you to define the rules which govern the robot's behavior and the results can be amazingly complex. It was a long and winding road from the initial spark of an idea to the finished product, a journey the author shares with us here.

For as long as I can remember I've always been fascinated by the idea of autonomous machines, particularly free-roaming robots able to interact with their environment. When I first started experimenting I would use FischerTechnik to make the models. My designs got more ambitious and when in 1993 I decided to make a robotic spider with the ability to move realistically while using the minimum amount of mechatronic hardware I realized I had reached the limit in terms of structural stability with this type of building material. I went ahead and decided to hand make a new spider body from wood with aluminum components for the legs, even the transmission was a home brew affair.

I had some basic knowledge of microcontrollers but in those days the likes of Arduino and Raspberry Pi were still to make an appearance. My only option for the controller was to go for hard-wired logic. I was able to build the control unit as part of a project at school. Using a binary counter I was able to make the spider move using pseudo random movements. Without any control input the spider would move, change direction and rest for random periods. The antennae and body would also waggle at random. Infra red light barriers were used to detect obstacles. When an obstacle was encountered the spider would pause, back up, turn and head off in the other direction. The combination of function and apparently random decisions gave the impression of some form of intelligent life form. My mother helped me make a covering for the robot so it looked like a giant tarantula. The model shown in **Figure 1** had a span of 60 cm!

With this and future developments I came to build robots not to perform specific functions but instead to explore the question; what is it about the behavior of a creature that convinces us its movements are a result of intelligence? The follow up, of course is how we can efficiently model this sort of behavior using the minimum hardware.

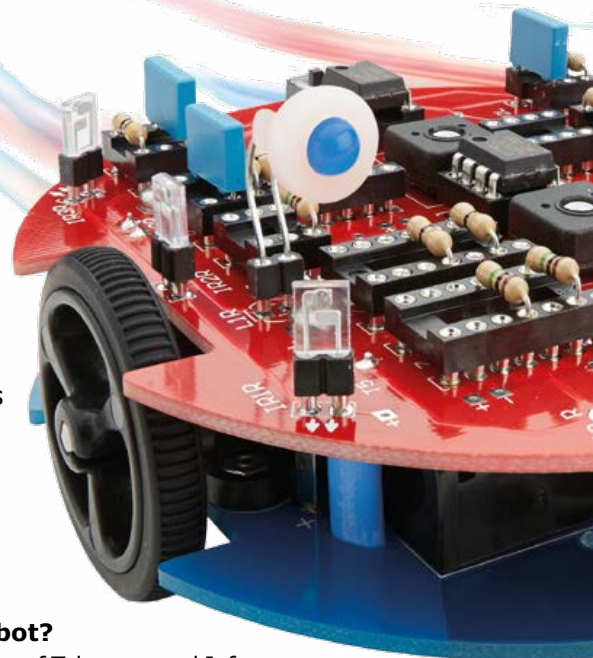
An analog controlled robot?

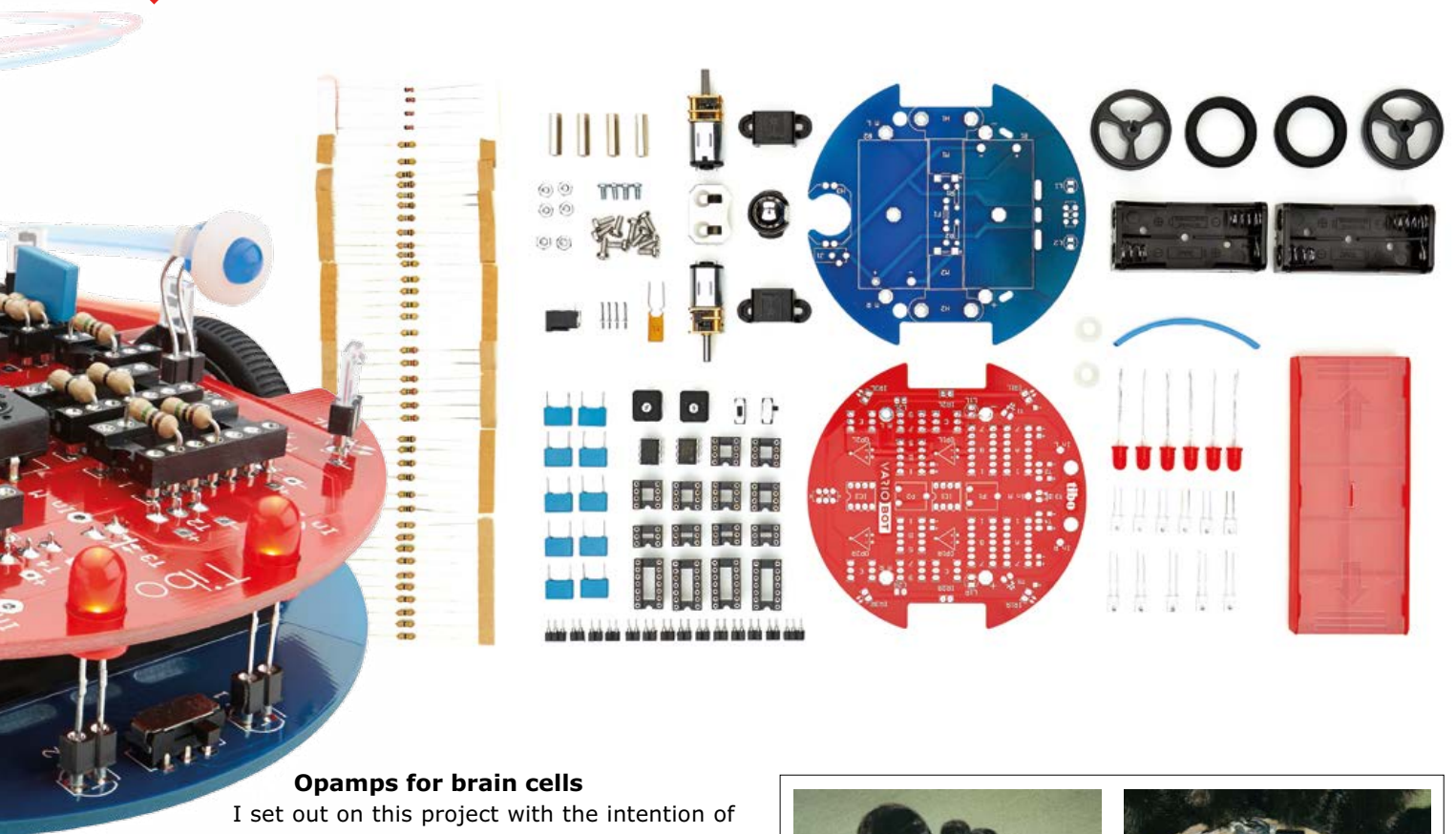
During my studies of Telecoms and Informatics in 1999 I was introduced to the concept of the Braitenberg vehicle [1]. These experiments describe how simple interconnections between sensors and propulsion motors can result in unexpectedly complex, apparently intelligent, behavior in a vehicle. I found both the simplicity of the platform and the complexity of the resulting movements fascinating. I began developing and optimizing many variants of the basic model, experimenting with the sensor integration and analog signal conditioning from the outputs of light sensors fitted to simple wheeled robots.

The main innovation with these models was the combination of groups of sensors connected in series. Unlike standard sensor outputs which provide absolute measurement values these supply more than n (2 or more) light sensors up to $n - 1$ relative values of brightness to navigate around obstacles. Our own ability to perceive objects is largely dependant on the background levels of brightness; this effect can be the cause of many optical illusions.

The simple transistor circuit shown in **Figure 2** shows the principal of this type of control for a robot with differential transmission which is directed to the right by sensor S1 and to the left by S2. The robot navigates according to relative brightness levels in the environment rather than any preset threshold. When the light sensors are fitted so that light does not fall directly on them, the robot dodges to avoid obstacles because the shadows produced reduces light to the sensor. A sensor placed between S1 and S2 will make the robot stop when it encounters an obstruction ahead.

With the sensors pointed down at the floor, this simple circuit functions as an effective line follower. To follow a dark line on a light background it will be necessary to swap the sensors S1 and S2. To replicate this circuit you can use Darlington transistors, these have enough gain to drive the motors directly-





Opamps for brain cells

I set out on this project with the intention of using operational amplifiers (opamps) as the basic analog brain cell to provide autonomous control. Power opamps can not only handle high impedance analog sensor signals but also provide bidirectional drive for the motors. At the time I used the L272 dual power opamp which is able to supply an output current of up to 1 A with a supply rail as low as 4 V, but a voltage level this low gives a relatively restricted range of control. In its simplest form a 'direct connection' between sensor and motor can be achieved with a impedance converter (voltage follower) where the output signal is fed back to the inverting input of the amplifier. Using symmetrical supplies a single opamp can drive the motor in both directions to give the reversing capability of Braitenberg vehicles.

As well as providing simple linear amplification an opamp can also be used to provide inversion, (weighted) addition or subtraction, logarithmic or comparator function where the level of two signals or their value compared to a threshold, will cause the output to change state. In the days of analog computers these arithmetic operations could be programmed via a plug board and I used a similar method in my model but this time using components plugged into IC sockets to program the functions and parameters of the required analog operations. My first and simplest vehicle consisted of two driving wheels, a swiveling castor and rechargeable batteries. A long DIP socket housed the dual opamp, and provided free sockets so that capacitors and resistors can be plugged in as necessary (Figure 3).

The series development

Back in 2013 when eight years of research had come to an end, I decided to patent my ideas, build a new generation of robots and market them. I had little idea at the time just how difficult the journey ahead would be. One of the greatest challenges was to design an experimental analog platform that was

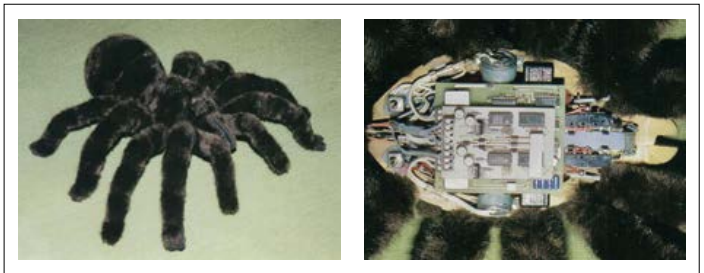


Figure 1. My first robot spider.

both simple yet highly flexible. After much tinkering with the PCB design I developed a circular red board containing control elements which can be seen at the top in Figure 4. The design makes it easy to access when plugging in components to define the control behavior for the suggested projects and

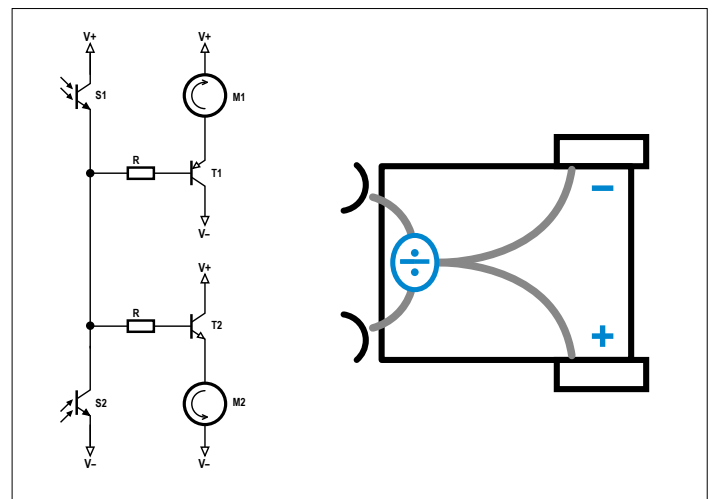


Figure 2. Simple transistor circuit for a robot with differential drive.

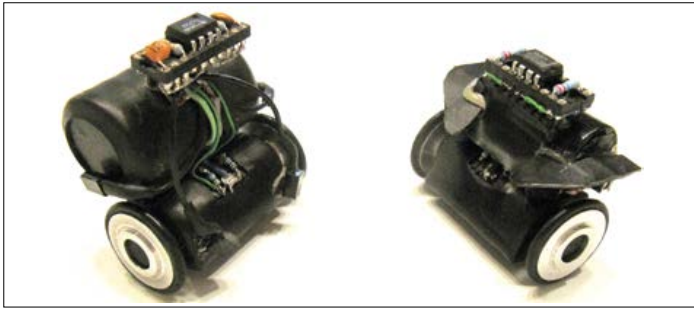


Figure 3. The first simple analog robot model.

also for free ad hoc experimentation. The second circular blue PCB (lower photo) fits into the base of the robot and houses Tibo's power source and motors.

In addition to the practicalities of the robot design it was also necessary to invest much time and thought in the preparation of a constructor's manual for the kit, introducing new didactic principles. Here the relationship between input signals and output variables are represented using bar graphs which gives an intuitive level of understanding without the need to resort to formulae. **Figure 5** shows an experimental configuration to build a line follower.

All of the components used in the design of Tibo are selected to ensure they are robust, reliable and easy to use. Low quality

materials such as cable ties and self adhesive fittings often found on low-cost construction kits were not even considered. I totally underestimated the hassle I would experience procuring and bargaining with all the different suppliers of all of the components.

The process of researching, testing and sourcing a suitable opamp took many weeks. For the kit of parts we planned to use just one type of opamp with a DIP package outline and this restricts our choice right from the start. As well as the motors, the opamp also drives the infrared LEDs. To avoid overloading them it's necessary to limit the current. Tibo gets its energy from four AAA sized primary or rechargeable cells which provide a supply ranging from 4 to 6 V (i.e. a symmetrical ± 2 V to ± 3 V) so a simple series resistor will not be suitable in this case. To keep the circuit both simple and straightforward it was decided not to use any additional type of IC such as a voltage regulator. The choice of the TS922 dual opamp has the advantage of an internal current limit. Using this opamp we can drive the IR LEDs (IR908-7C-F) directly without any risk of overdriving them. This feature also protects the motors, should Tibo become trapped somewhere.

Component sourcing

With all the development work complete and the majority of opamps already ordered I discovered that the TS922IN had been discontinued in its DIP8 package by STMicroelectronics.

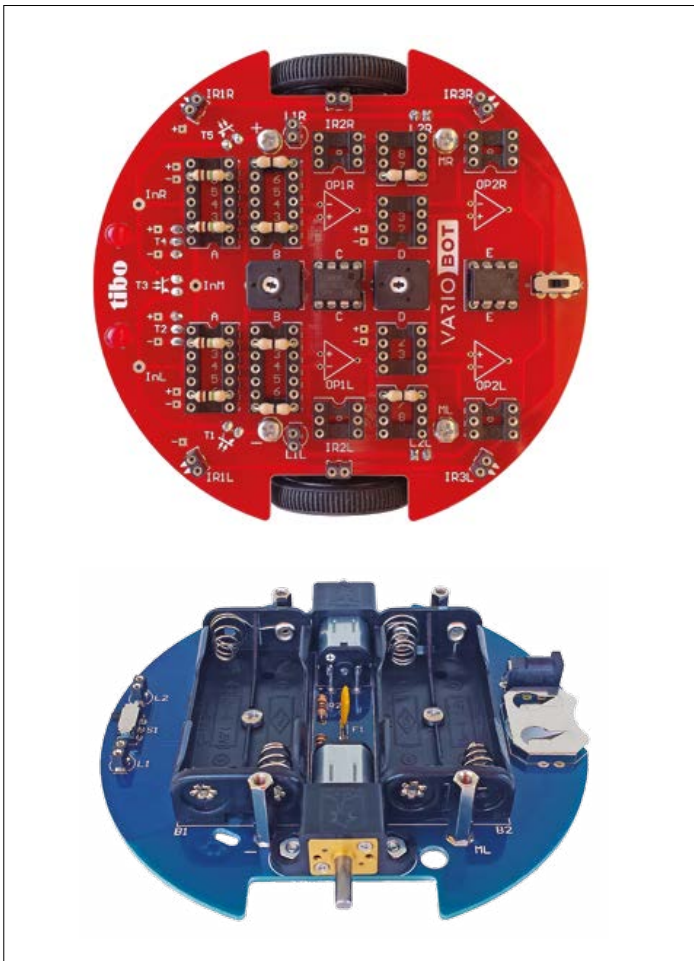


Figure 4. Tibo's control PCB (left) and power PCB (right).

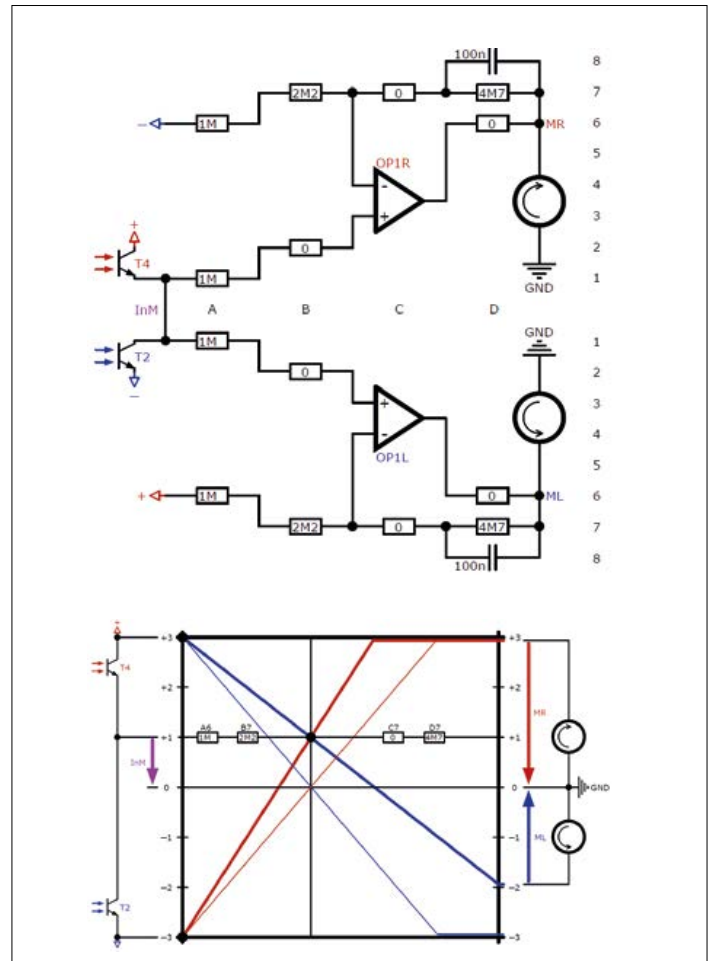


Figure 5. An example from the experimenter's instruction manual.

I was short of 300 of these ICs so started out on a quest to track down any remaining stocks on the retailer's shelves but ended up sourcing some from the Far East. These unfortunately turned out to be counterfeit and could only deliver around 12 mA instead of the expected 80 mA. I made 30 requests to tender, supplying information such as the complete part number together with photos of the exact chip and ended up placing over ten trial orders to test the components but all were unsuccessful. All of the samples had exactly the same fault. After much fruitless searching I discovered the German company TronicPool who specialize in sourcing discontinued components. They were successful in tracking down the last piece of the jigsaw; at last my quest was at an end.

Sourcing the metal-g geared motors (PGM-12F) was also just as protracted. These particular motors measuring 25 x 12 x 10 mm³ are quite compact and available in a range of operating voltage, gear ratio and power. The standard operating voltage seems to be 6 V but 3 V versions (as used in Tibo) are also available. On the grounds of costs, it was not possible to source these from one European supplier. You can expect to pay around €15 per unit for a low volume order. There are many suppliers on eBay such as Aliexpress and Alibaba offering these types of motors for a very good price. You need to exercise caution here though, sometimes you can receive a perfectly acceptable sample but that's no guarantee that future supplies will be of the same quality. While you can expect low delivery costs you need to take into account delivery times which are typically a few weeks longer than from local suppliers. During tests I tried out a total of seven different types of motor for the robot. The properties I was looking for were low power consumption, low start up voltage and smooth running. Many of the examples failed because of wobbly spindles and failure to even begin rotating. For personal use and for small volume orders I would recommend using motors sourced from local suppliers, in the long run you have better quality control and will spend less time fielding customer complaints.

And more problems...

Anyone toying with the idea of developing and bringing a product to market should be aware that as well as the technical challenges there are also many other aspects of the process that will take up your time and money.

I had already decided to patent both the controller and the kinematics concepts of the robotic spider. All the necessary research proved very interesting and I could compare alternative ideas and identify the essence of my own design that could be protected. Owning the rights to your Intellectual Property (IP) is of course no guarantee that your product will be marketable or successful but I recommend the use of a Patent library or patent attorney to ensure your idea does not infringe other designs already protected. The same goes for the registering of a trade mark. In Germany we have the government funded SIGNO institution which supports small and medium sized businesses as well as individuals, offering advice on legal protection and commercial exploitation of innovative ideas. Check the government web sites active in your own country for similar schemes.

Finally for products manufactured in the EU there is CE conformity. This mark is your declaration that the product meets all

the requirements of the applicable EC directives. It is important to check out any legislation concerning the product life cycle, eventual recycling and even packaging made from biodegradable products. When in doubt, it is advisable to seek out the help of professionals even if you think initially that you will be able to deal with all of the matters yourself. Just to get some experience I started to put together the first 100 kits but quickly realized it's better to leave such tasks to the professionals. After this initial phase all subsequent kits have been expertly tested and packed by a local workshop.

For many inventors it may be a better solution to license your invention so that all production worries are taken over by an established manufacturer. This way you receive an agreed percentage of the sales and you don't incur the risks associated with forming your own company. There can also be pitfalls if you choose this path but instead of all the stress and time involved in marketing your own product you can concentrate on designing the next one. ◀

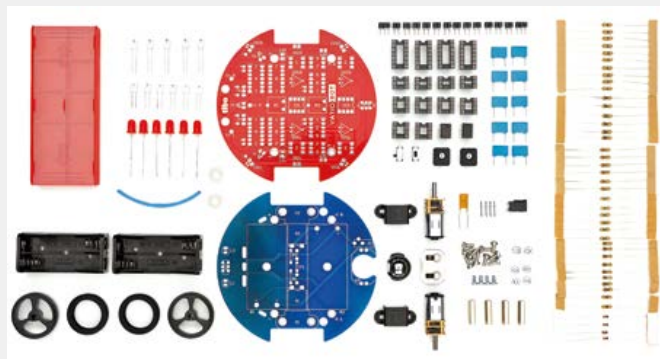
(140510)

[1] Braitenberg, Valentino: *Vehicles: Experiments in Synthetic Psychology* (MIT Press 1986)

Tibo from Elektor Store

The Elektor Store provides a fantastic opportunity to present fascinating developments such as the Tibo robot kit to customers worldwide. The kit of parts contains all the mechanical and electronic components necessary to build Tibo.

At www.elektor.com/tibo-robot-kit you will also find more information about the kit and ordering information. In addition there is a 40-page builders and experimenters guide and a video showing construction of the robot in detail.



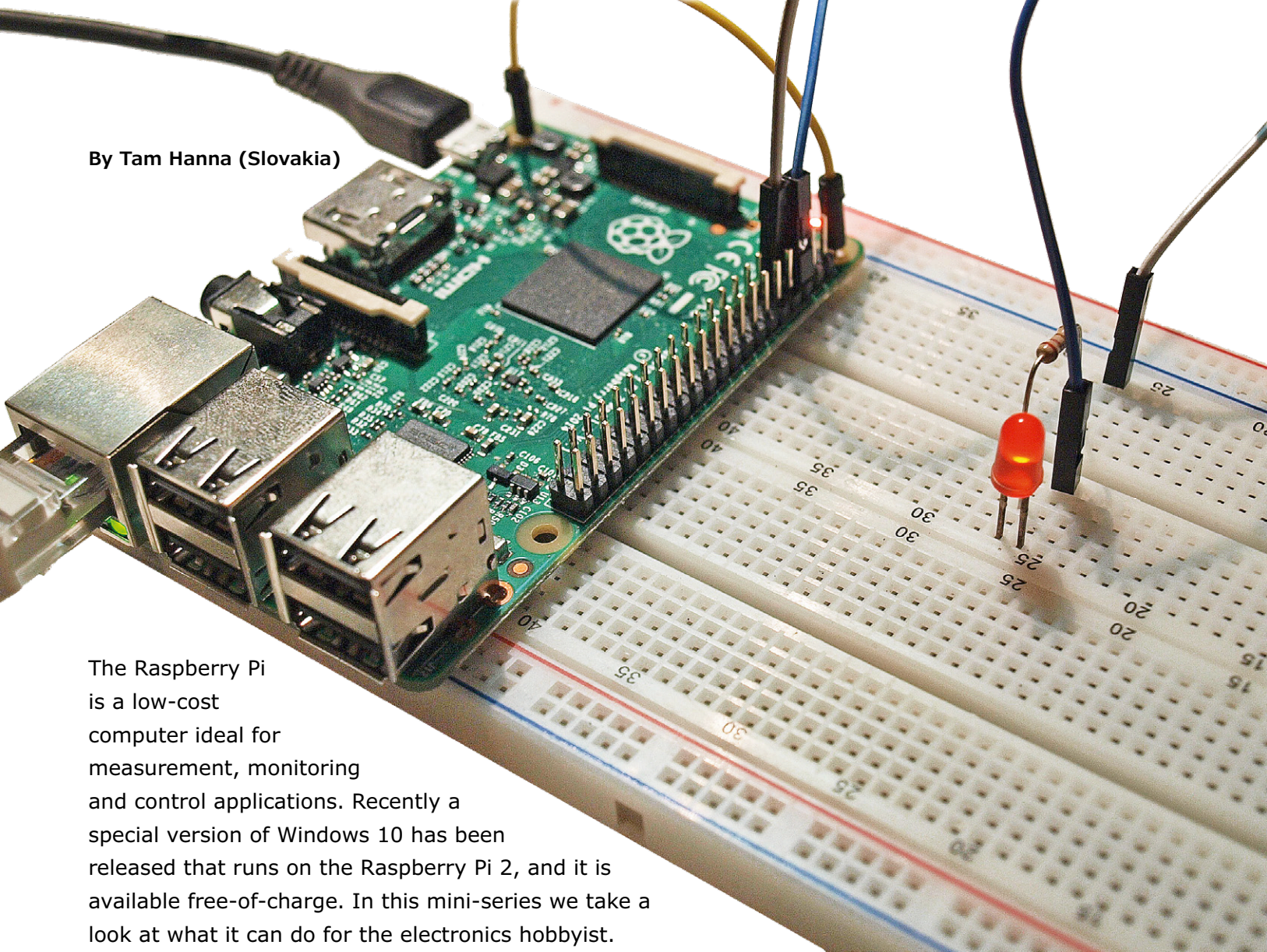
The kit costs €89.95

Elektor members grab a 10% discount!

Windows on the Raspberry Pi (1)

Installation and first programs

By Tam Hanna (Slovakia)



The Raspberry Pi is a low-cost computer ideal for measurement, monitoring and control applications. Recently a special version of Windows 10 has been released that runs on the Raspberry Pi 2, and it is available free-of-charge. In this mini-series we take a look at what it can do for the electronics hobbyist.

Whether you are a fan or not, there is no denying that British entrepreneur Eben Upton's company has made a powerful single-board computer that is affordable by almost anyone. And now, thanks to a partnership with Microsoft, it is possible to run a version of Windows 10 on the Raspberry Pi 2. The version in question, Windows 10 IoT Core, is by no means a desktop operating system, and it will not turn your Raspberry Pi 2 into a desktop replacement. It is rather a trimmed-down version of Windows 10 designed for use in 'kiosk mode'. This means that the operating system will only ever run one specified application while the system is on: for example, this might be a measurement, monitoring or control application, in which case we might think of the machine as a dedicated process control computer.

First steps

If you want to run Windows 10 on your Raspberry Pi 2, you will need a desktop computer to develop programs for it. That computer must run Windows 10: if your desktop is currently running Windows 7 or Windows 8 you will therefore need to carry out the (free) upgrade to Windows 10. The operating system for the Raspberry Pi is provided in a 'container' format newly-introduced with Windows 10, which cannot directly be transferred to an SD card (minimum 8 GB capacity required) with older versions of Windows.

Navigate your way to the website at [1] and click on the download link in the section headed 'Windows 10 IoT Core for Raspberry Pi 2' to download the image file. In the Edge browser you

▶ Advanced debugger functions simplify finding problems

can now open the ISO file by clicking on 'Open' in Explorer, which will cause the installation program contained in the image to start running.

Microsoft has simplified the deployment of the operating system with a tool called 'WindowsIoTImageHelper'. In the first step the application asks you to select the card reader device to be used, and in the second step you must select the image file at C:\Program Files (x86)\Microsoft IoT\FFU\RaspberryPi2\flash.ffu. Clicking on the 'Flash' button begins the process of transferring the operating system to the SD card, which can take several minutes.

Now plug the SD card into the Raspberry Pi 2, which should also have an Ethernet connection, as well as keyboard, mouse and monitor attached. The first boot-up typically seems to take about ten minutes. After the obligatory reboot the Raspberry Pi's 'desktop' should appear on the screen: this is a dedicated application which just displays the IP address of the machine. If the Raspberry Pi is used without a screen ('headless'), the IP address can usually be discovered via the web interface of the router it is connected to. In our case the Raspberry Pi can be found at IP address 192.168.0.100.

The development environment that we need is Visual Studio 2015. The free 'community' version can be downloaded from [2] and installed: it is worth doing this installation in parallel with downloading the ISO image. Make sure that you choose the 'Custom' option in the installation wizard; in the next step you will need to select the option 'Universal Windows App Development Tools'.

Before we can develop an application for the Raspberry Pi 2 we have to extend Visual Studio to include the necessary project templates. This can be done using the extension manager built into the IDE, accessed under Tools - Extensions and Updates. Click on the 'Online' column and then in the 'Visual Studio Gallery' column look for 'Windows IoT Core Project Templates'. A click on 'Download' will cause these to be loaded automatically onto your machine.

Windows 10 workstations have to be put in developer mode before they can be used for programming. Under 'Settings' click on Update & Security - For developers - Developer mode.

We start with a skeleton project

Visual Studio 2015 is not limited to creating Raspberry Pi applications: it can also build programs for Windows PCs and even Windows smartphones.

After clicking on 'New Project...' a wizard appears, in which you should select the template 'Visual C# - Windows - Universal - Blank App (Universal Windows)'. This will create an application for the WinRT runtime environment, which is supported across all recent Microsoft operating systems. For the purposes of this article we shall call our first program 'ElektorRPI', but you can of course choose any name you wish.

Visual Studio programs are called 'solutions', and can contain several independent projects. When debugging is initiated the project shown in bold face will be the one that is run. To change this, use a right mouse click and 'Set Startup Project'. In the case of our skeleton project, however, we have only a single Universal Windows application.

Applications' user interfaces are described using a modern technology called XAML, a derivative of XML. Each XAML file consists of the mark-up itself (which gives the layout of control elements such as buttons and text entry fields), and is accompanied by a 'code-behind' file, which contains the required program code. The code is conventionally written in the C# .NET language, but development in Visual Basic is also possible for the Raspberry Pi.

Developers with experience of writing for microcontrollers may not be used to the event-driven paradigm that is used when writing Windows 10 applications. This means that there is no main loop to be seen in the program code, and no polling of buttons or inputs. Instead the vast majority of the code comprises functions which are called in response to specified events, called 'event handlers'. Among these event handlers might be a function that responds to a button in the user interface being clicked; another type of event is generated when a character is received.

Running our first program

At the moment our application does nothing more than present a black screen. This is not a problem, as the program is still under the control of the debugger process.

Figure 1 shows the part of the Visual Studio toolbar that relates to configuring the debugger. The first combo box allows the

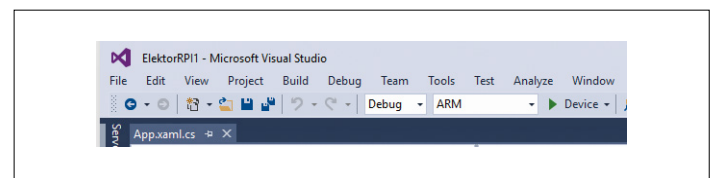


Figure 1. Debugger configuration.

compilation mode to be chosen; the second is used to specify the target system processor architecture. For the Raspberry Pi 2 this should be set to 'ARM'.

The button with the 'play' symbol is divided into two parts. The small downward-pointing arrow opens a context menu to allow selection of the target system. Click here and select 'Remote machine'. This means that the application will not be run locally under Visual Studio on the development computer but instead remotely on the Raspberry Pi, using its network con-

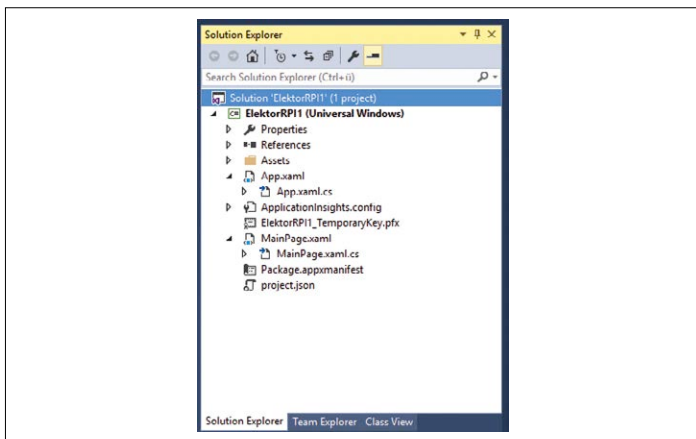


Figure 2. Visual Studio projects are comparatively complex.

nection. The IP address of the Raspberry Pi should be entered in the 'Address' field in the pop-up window that appears, and for 'Authentication mode' select 'None'. Now click on 'Select' to save these settings. You can change these settings later by right-clicking on the project (that is, the universal Windows application) in the 'Solution Explorer' as shown in **Figure 2** and then selecting Properties - Debugging.

The deployment process itself is carried out without further prompting by clicking on the play symbol. When first transferring a program Visual Studio starts by sending a few libraries to the target machine, a process which can take up to a minute. Then the status line at the bottom of the screen will change to a dark orange, which means that the program is being executed.

Bug-hunting using Visual Studio

So far we have yet to receive any message back from the target machine. We address that now by causing the Raspberry Pi to send a brief text string to the debugger console. This will happen when the program starts up, when the target loads its 'MainPage', the root window of the application. The C# language is object-oriented and so windows themselves are also objects, which must be initialized using a 'constructor'. The constructor for the MainPage is generated automatically by Visual Studio when the project is set up, and can be found in the code-behind file MainPage.xaml.cs. If some special action is required when a program starts, the necessary commands can be added to the MainPage constructor, which is always called before the application window appears (see **Listing 1**).

Listing 1. The Raspberry Pi sends us a message when the program starts.

```
public sealed partial class MainPage : Page{
    public MainPage() {
        this.InitializeComponent();
        System.Diagnostics.Debug.WriteLine("System started");
    }
}
```

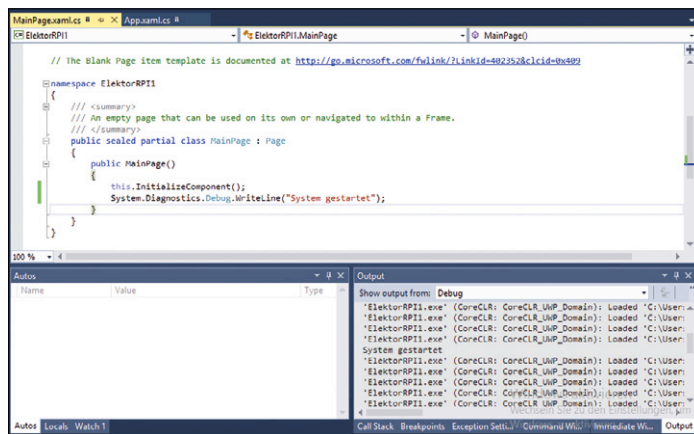


Figure 3. At the bottom right is the text message output by our program.

Let us now try this out by clicking on 'Play'. The Visual Studio screen reconfigures itself when a program is run: for reasons the author cannot fathom the output window only appears in the edit view. The problem can be fixed by changing the setting at View - Output, to cause the text message to appear among the runtime messages: see **Figure 3** at the bottom right.

Tracking down problems is made easier with the help of advanced debugging functions. Double-clicking on the gray area just to the left of the source code allows a breakpoint to be set: the debugger will now halt program execution whenever the line of code in question is reached. In this state of suspended animation you can inspect the values of local and global variables, and then resume program execution either one step at a time or continuously.

GPIOs

The header provided on the single-board computer allows peripherals to be connected. Most simple control applications can be handled using the part of the connector that features the GPIO pins, which behave in the same way as their namesakes on PICs, AVR and the like. Watch out, however, because the Broadcom processor does not compete on robustness with these smaller microcontrollers: the signal level is 3.3 V, and neither inputs nor outputs are 5 V-compatible. In calculating the overall power budget the designers of the Raspberry Pi reckoned on a maximum current of 3 mA being drawn from each pin, and it is advisable not to exceed this figure.

For our first program that is actually capable of controlling something from the Raspberry Pi running Windows, we will connect an LED with series current-limit resistor between a GPIO pin (GPIO 4 in this case) and ground. This mini-circuit is shown in **Figure 4**.

Programming

The idea behind the object-oriented programming language C# is that elements of the real world are represented as objects which contain data and code. An object of class GpioPin is used to represent a GPIO pin. More advanced programmers can take

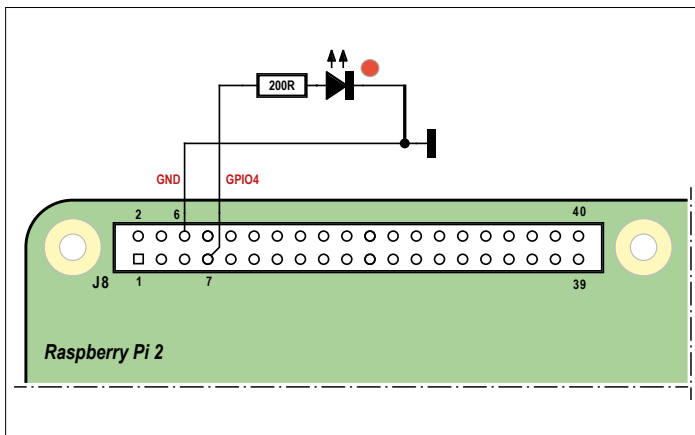


Figure 4. An LED is connected to GPIO pin 4 via a series current-limiting resistor.

a look at the declaration of this class, an excerpt from which appears in **Listing 2**. At a glance you can see what ‘methods’ are available in the class: the names of the methods should be self-explanatory.

Some readers may have heard of ‘garbage collection’, a process where Windows from time to time deletes any unused objects in order to reclaim the storage space they occupy. This applies also to Windows on the Raspberry Pi and its `GpioPin` objects: in this case, when the object is deleted the corresponding hardware pin is automatically returned to the high impedance state. This can be a problem in electronic projects, which we shall investigate more closely in the third part of this series. To obtain access to a GPIO pin we must first ‘open’ it. This is done using the `OpenPin` method of an object of class `GpioController`. This object is created using the line

```
GpioController myGPIO = GpioController.GetDefault();
```

and then pin X is opened using

```
GpioPin myPin = myGPIO.OpenPin(X);
```

which also returns an object of class `GpioPin`. In this case we have called the object `myPin`.

Readers who have already done some programming using the .NET framework will know that before using a class the corresponding ‘namespace’ should be declared at the start of the code file, with a ‘using’ declaration. Missing declarations are indicated by Visual Studio underlining the affected elements. A context menu (accessed using a right mouse click) helps to identify the correct namespace. The class `GpioController`, however, cannot be located in this way because it is in the form of an extension.

To solve this problem click on the ‘References’ folder in the solution with the right mouse button, and from the context menu that appears select ‘Add reference’. Then switch to Universal Windows - Extensions and add a tick before the package

Listing 2. The `GpioPin` class and its most important methods.

```
namespace Windows.Devices.Gpio
{
    public sealed class GpioPin : IGpioPin, IDisposable
    {
        public GpioPinValue Read();
        public void SetDriveMode(GpioPinDriveMode value);

        public void Write(GpioPinValue value);

        ...
    }
}
```

‘Windows IoT Extensions for the UWP’. The `GpioController` class will now appear in the context menu.

Beginners may find dealing with the two classes `GpioController` and `GpioPin` somewhat confusing at first. As usual, a good way to gain experience with such things is to start with an example program written by someone else, and then try modifying it one step at a time.

Lighting an LED

Now we come to our demonstration. As usual, the code is available for free download at [3]. First we create an object of class `GpioController` and then open the pin. Next we configure the pin as an output using the method `SetDriveMode()`. Finally we set the output high:

```
GpioController myGPIO = GpioController.GetDefault();
GpioPin myLEDPin = myGPIO.OpenPin(4);
myLEDPin.SetDriveMode(GpioPinDriveMode.Output);
myLEDPin.Write(GpioPinValue.High);
```

As described above, we can include this code in the `MainPage` constructor so that it will be executed when the program starts up.

Try it out, and you should be greeted by an illuminated LED!

Now to make it blink

With a minimum of extra effort we can extend the program so that the LED blinks continuously. The necessary code is shown in **Listing 3**. As you can see, the code is again inserted into the `MainPage` constructor so that it is executed when the program starts. The first part of the code we have seen already; the section that follows initializes a timer with a period of one second, and starts it running. An interesting point is the line

```
myTimer.Tick += MyTimer_Tick;
```

which supplies a function to the variable `Tick` which belongs to our object `myTimer`. The function will be called when the timer interval expires, that is, once per second. The function

MyTimer_Tick is our event handler, whose code is also included in the code file MainPage.xaml.cs (see the end of Listing 3). Understanding these lines should present no difficulty to anyone who has already had a little experience in programming microcontrollers using any dialect of C.

Why do we need to use a timer here rather than simply writing the usual delays inside an infinite loop? Unfortunately, what is common practice in the world of eight-bit microcontrollers is here strictly verboten: WinRT programs are not allowed to block in CPU-time-consuming activities as this would prevent the user interface from functioning. We will look at this aspect in more detail in the second part of this series.

Speed

As electronics engineers we are of course interested in seeing how quickly we can make the LED blink. The class TimeSpan puts one limit on what is possible: the smallest possible time span that can be represented is one millisecond.

```
myTimer.Interval = new TimeSpan(0, 0, 0, 0, 1);
```

If the program is run with this modification, it should be built using the 'Release' compilation mode. This is because allowing access by the debugger costs additional processor time on the Raspberry Pi.

If we now connect an oscilloscope to the pin, we will unfortunately see a rather ugly waveform. The pulse width changes erratically because the code execution takes too long. **Figure 5** shows an example trace which indicates that the average execution time is around 30 milliseconds.

Note that simplifying the loop along the following lines does not work:

```
private void MyTimer_Tick(object sender, object e)
{
    myLEDPin.Write(GpioPinValue.High);
    myLEDPin.Write(GpioPinValue.Low);
}
```

This is because the 'Write' commands only update the value stored for the pin within the GPIO driver. The level on the pin itself is only updated after the timer handler completes.

Conclusion

The Raspberry Pi 2 may be cheap, but it is nevertheless a fully-featured computer ideal for measurement, monitoring and control applications. However, in comparison to classical microcontrollers there is a certain amount of overhead in programming it.

As compensation for this additional overhead, it is possible to implement graphical and networking functions with very little bureaucracy. In the next part of this series we will let our single-board computer communicate with the outside world using TCP/IP. ◀

(150465)

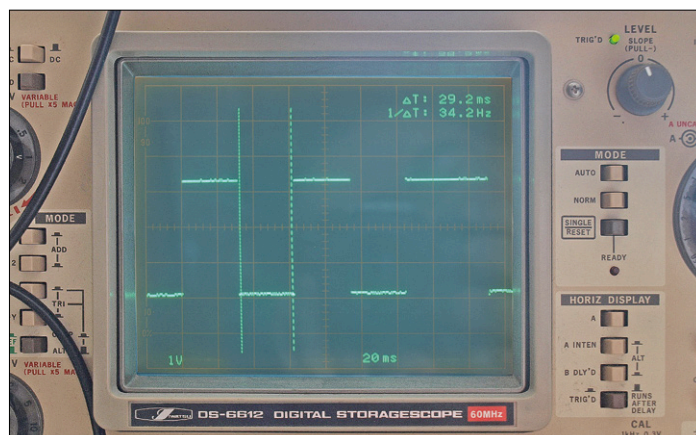


Figure 5. Not a pretty sight: oscilloscope trace when attempting to switch the output every millisecond.

Web Links

- [1] <http://ms-iot.github.io/content/en-US/Downloads.htm#Win8>
- [2] www.visualstudio.com/de-de/products/visual-studio-community-vs.aspx
- [3] www.elektormagazine.com/150465

Listing 3. A blinking LED.

```
GpioPin myLEDPin;
bool curVal = false;

public MainPage()
{
    this.InitializeComponent();

    GpioController myGPIO = GpioController.GetDefault();
    myLEDPin = myGPIO.OpenPin(4);
    myLEDPin.SetDriveMode(GpioPinDriveMode.Output);

    DispatcherTimer myTimer = new DispatcherTimer();
    myTimer.Interval = new TimeSpan(0, 0, 1); // 1 s
    myTimer.Tick += MyTimer_Tick;
    myTimer.Start();
}

private void MyTimer_Tick(object sender, object e)
{
    curVal = !curVal;
    if (curVal)
        myLEDPin.Write(GpioPinValue.High);
    else
        myLEDPin.Write(GpioPinValue.Low);
}
```

Professional Quality @ Discount Prices!

reichelt.co.uk
elektronik



- ✓ More than 45 years of experience
- ✓ 24-hour shipping
- ✓ More than 50,000 products



onlineshop languages:



Druckluft 67 3,550 € / 100 ml

KONTAKT 334 400 ml **14,20**
(~ 10,30 GBP)

Kälte 75 3,550 € / 100 ml

De-icer spray down to -52 °C

KONTAKT 317 400 ml **14,20**
(~ 10,30 GBP)

Kontakt 60 2,525 € / 100 ml

Oxide-dissolving contact cleaner

KONTAKT 203 400 ml **10,10**
(~ 7,33 GBP)

Kontakt WL 2,650 € / 100 ml

Universal cleaner

KONTAKT 208 200 ml **5,30**
(~ 3,85 GBP)

Sprühöl 88 3,100 € / 100 ml

Mechanic's oil

KONTAKT 230 200 ml **6,20**
(~ 4,49 GBP)

Solvent 50 3,050 € / 100 ml

Label remover

KONTAKT 244 200 ml **6,10**
(~ 4,42 GBP)

Screen TFT 2,950 € / 100 ml

screen cleaner

KONTAKT 501 200 ml **5,90**
(~ 4,28 GBP)

SUBSCRIBE NOW!

Newsletter

Receive weekly fresh information about

- ✓ product innovations
- ✓ specials
- ✓ Price reductions



Daily prices | Price as of: 22.09.2015

Prices in € incl. statutory VAT, plus shipping costs
reichelt elektronik, Elektronikring 1, 26452 Sande (Germany)

A tool set to get excited about

22-piece set in a high-quality pouch

Compact ratchet and screwdriver set. Through careful selection of the most important tools, this combination optimally equips you for most jobs

- 1 Zyklop mini-ratchet, 7 sockets (8, 10, 12 and 13), 1 extension with free-turning sleeve
- 1 handle with quick-change chuck for 1/4" bits, 4 cross slot bits, 1 plain slot bit, 5 TORX bits, 4 hexagon bits
- Pouch with belt loop

WERA KK Z MINI

Limited special edition

49,00
(~ 35,55 GBP)

Wera Lasertip blade

Laser treatment produces a tip with a micro-precision surface and an edge with a Vickers hardness of up to 1000 for a firm grip on the screw head.

Plain slot
160 i VDE
1x 0.4x2,5x80,
1x 0.6x3.5x100,
1x 0.8x4.0x100,
1x 1.0x5.5x125

Cross slot
162 i PH VDE
1x PH 1x80,
1x PH 2x100

Voltage tester
247 1x 0.5x3.0x75

Set of plain slot and cross slot screwdrivers



Bestellnummer:

WERA 6147

29,95 (~ 21,73 GBP)

7-piece set + rack

Wera Tool-Check

Set includes 1 Rapidaptor bit holder and 1 Wera bit ratchet

- 28 bits
- 1 adapter
- 7 sockets



Bestellnummer:

WERA 05056490001

79,95
(~ 58,01 GBP)

Order now! www.reichelt.co.uk

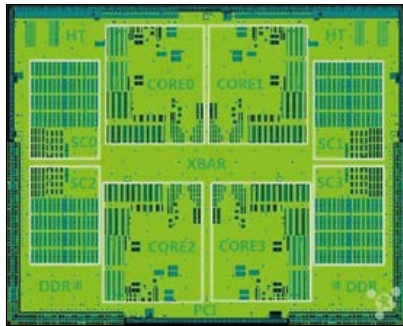
Order Hotline: +49 (0)4422 955-360

Payment Methods:





The ElektorBusiness section in Elektor Magazine accommodates articles, news items and other contributions from companies and institutions active in electronics. Publication is at the discretion of the Editor. ElektorBusiness Editor: Jan Buiting Contributions to: newsdesk@elektor.com



Chinese MIPS-based 64-bit CPUs run x86, ARM, Linux

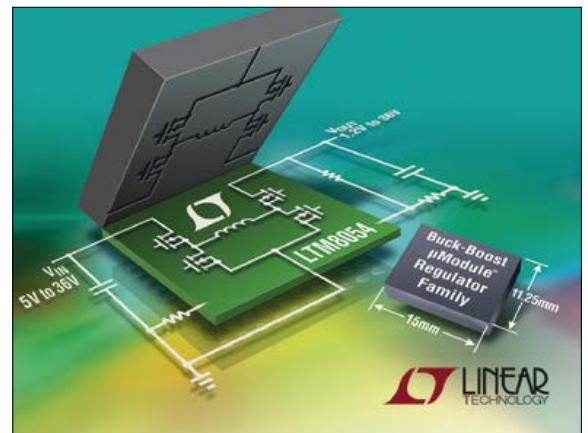
Loongson's 3A2000 and 3B2000 are two quad-core 64-bit processors based on an enhanced version of the MIPS64 architecture called LoongISA. These 4-way superscalar processors are built on a 9-stage, super-pipelined architecture with in-order execution units, two floating-point units, a memory management unit, and an innovative crossbar interconnect.

3A2000 is a flagship CPU aimed at the high-performance consumer electronics market (e.g. desktop computers and laptops, 64-bit embedded and DSP applications, and network routers) while 3B2000 will be used in a number of home-grown eight and 16-core server systems. Remarkably, the two CPUs can also execute x86 and ARM code using LoongBT, a binary translation technology for Linux. Benchmarking data released recently shows the MIPS64-based powerhouse CPUs surpassing several competing processors in performance efficiency (SPEC CPU2000 per GHz), including ARM Cortex-A57 and AMD E1-2100.

www.loongson.cn (150486-3)

5.4 A, 36 V Buck-Boost Regulator

Linear Technology's LTM8054, 36V_{IN} (40 V_{max}) 5.4A μModule® (micromodule) regulator in small 11.25mm x 15mm x 3.42mm BGA package regulates an output voltage equal to, greater or less than the input voltage. This enables a regulated output voltage even if either or both input and output voltages vary greatly as the device seamlessly transitions between buck and boost functions. The device includes the inductor, DC/DC regulator, MOSFETs and the supporting components. The LTM8054 regulates an output voltage between 1.2 V and 36 V, from 5 V to 36 V input supply in three operating modes of buck (step-down), boost (step-up) and buck-boost (V_{IN} approximates V_{OUT}). The LTM8054 includes a 4-switch buck-boost topology to maximize efficiency: 92% 24V_{IN}, 12V_{OUT} (buck) and 94% 36V_{IN}, 24V_{OUT}. Transition between the modes of operation is automatic, making the LTM8054 a simple and compact DC/DC regulator solution for a wide range of applications such as industrial controls, automotive, solar and high power battery-supplied systems.



The LTM8054 has input and output current limit and monitoring functions. The switching frequency is adjustable by an external single resistor from 100 kHz to 800 kHz and it can be synchronized to an external clock from 200 kHz to 700 kHz.

www.linear.com/product/LTM8054 (150486-4)

Direct Conversion

CML Microcircuits' new CMX994A and CMX994E are RF receiver ICs featuring I/Q demodulators with low power consumption and high performance features. These devices are targeting the next generation of narrowband and wideband Software Defined Radios (SDR) for wireless data and two-way radio applications. Their design provides the optimum route for high integration, allowing a small RF receiver to be realized with a minimum of external components in zero-IF, near zero-IF and low-IF systems.

The CMX994A and CMX994E ICs build on the success of the popular CMX994 and are the first devices to use CML's PowerTrade™ technology. PowerTrade™ enables the devices to dynamically balance power consumption and performance characteristics to suit varying operating requirements. Very low power consumption can also be achieved in standby mode whilst looking for an RF signal, using intelligent control of power cycling, phase control and I/Q channel selection. The CMX994A delivers a very low power DCRx device while the CMX994E also includes the low power mode of the CMX994A but in addition,

offers a high performance mode with improved IP3 performance. The CMX994, CMX994A and CMX994E DCRx ICs offer excellent RF performance, exceptional IP2 from I/Q mixers and are suitable for modulation schemes including: QAM, 4FSK, GMSK and pi/4-DQPSK. Key features of the device include on-chip VCO for VHF applications, on-chip LNA, precision baseband filtering with selectable bandwidths and the smallest PCB area, typically less than 50% of a dual superhet.

www.cmlmicro.com (150486-2)



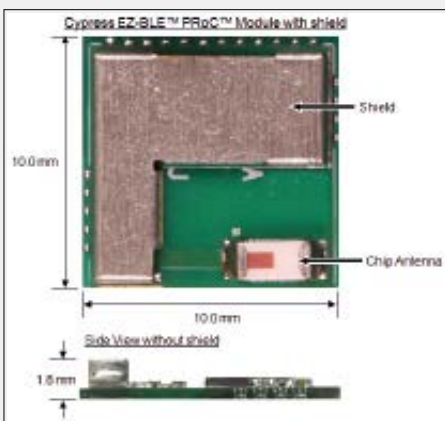
EZ-BLE™ PRoC™ Module = End-to-End Support

Cypress Semiconductor Corp.'s EZ-BLE PRoC module integrates the programmability and ARM® Cortex®-M0 core of PRoC BLE, two crystals,

an on-board chip antenna, metal shield and passive components, all in a compact 10-mm x 10-mm x 1.8-mm form factor. Customers designing with the module can apply to add the Bluetooth logo on their products by referring to Cypress's Qualification Design Identification (QDID) 67366, a unique serial number assigned by the Bluetooth SIG. Additionally, the module is compliant to wireless regulatory standards in the U.S., Canada, Japan, Korea and Europe. The module saves customer approximately \$200,000 in development, testing and certification costs.

Cypress has abstracted the Bluetooth Low Energy protocol stack and profile configuration into a royalty-free, GUI-based BLE component that can be dragged and dropped into designs using Cypress's PSoC® Creator™ integrated

design environment (IDE). PSoC Creator enables complete system design in a single tool.



Application details for Cypress's BLE Component are embedded in PSoC Creator with examples of all supported Bluetooth Low Energy profiles and hundreds of example projects for mixed-signal system designs. The \$49 BLE Pioneer Development Kit gives users easy access to the Cypress Bluetooth Low Energy devices, while maintaining the design footprint from the popular PSoC 4 Pioneer kit. The development kit includes a USB Bluetooth Low Energy dongle that pairs with the CySmart master emulation tool, converting a designer's Windows PC into a Bluetooth Low Energy debug environment. The EZ-BLE PRoC module can be quickly and easily evaluated with the EZ-BLE PRoC Module Evaluation Board, which plugs into the BLE Pioneer Development Kit.

www.cypress.com (150335-2)

Graphene gets Competition

Graphene, the only one atom thick carbon network, achieved overnight fame with the 2010 Nobel Prize. But now comes competition: Such layers can also be formed by black phosphorous. Chemists at the Technical University of Munich (TUM) have now developed a semiconducting material in which individual phosphorus atoms are replaced by arsenic. In a collaborative international effort, American colleagues have built the first field-effect transistors from the new material.

For many decades silicon has formed the basis of modern electronics. To date silicon technology could provide ever tinier transistors for smaller and smaller devices. But the size of silicon transistors is reaching its physical limit. Also, consumers would like to have flexible devices, devices that can be incorporated into clothing and the likes. However, silicon is hard and brittle. All this has triggered a race for new materials that might one day replace silicon. Black arsenic phosphorus might be such a material. Like graphene, which consists

of a single layer of carbon atoms, it forms extremely thin layers. The array of possible applications ranges from transistors and sensors to mechanically flexible semiconduc-



tor devices. Unlike graphene, whose electronic properties are similar to those of metals, black arsenic phosphorus behaves like a semiconductor.

A cooperation between the Technical University of Munich and the University of Regensburg on

the German side and the University of Southern California (USC) and Yale University in the United States has now, for the first time, produced a field effect transistor made of black arsenic phosphorus.

The compounds were synthesized by Marianne Koepf at the laboratory of the research group for Synthesis and Characterization of Innovative Materials at the TUM. The field effect transistors were built and characterized by a group headed by Professor Zhou and Dr. Liu at the Department of Electrical Engineering at USC.

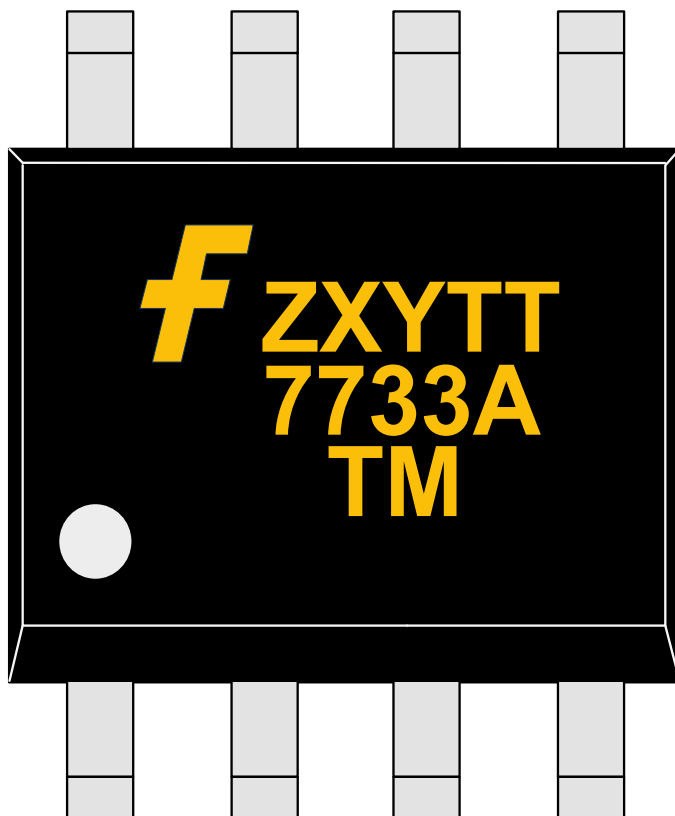
This work was supported by the Office of Naval Research (ONR), the Air Force Office of Scientific Research (AFOSR), the Center of Excellence for Nanotechnologies (CEGN) of King Abdul-Aziz City for Science and Technology

(KACST), the German Research Council (DFG) and the TUM Graduate School.

<http://www.acinomat.ch.tum.de> (150335-5)

Publication:

Black Arsenic-Phosphorus: Layered Anisotropic Infrared Semiconductors with Highly Tunable Compositions and Properties
Adv. Mater., 2015, Early View – DOI: 10.1002/adma.201501758
<http://onlinelibrary.wiley.com/wol1/doi/10.1002/adma.201501758/abstract>



LED Driver

with High PF and Ultra Wide Output Voltage

By Inki Park



As LED lamps are used in a variety of applications, they come in various shapes and sizes to fit specific fixtures. Consequently LED drivers should be designed reasonably to be suitable for the various LED types as well as customized specifications. Consequently we need to look at the PF and THD aspects of the design.

An LED voltage may vary considerably depending on the number of LEDs in series, and temperature. LED current however should remain constant no matter which type of design because the total lumen is proportional to the current. At the same time, high Power Factor (PF) and low Total Harmonic Distortion (THD) have become key design requirements for LED drivers. Therefore, a driver suitable for wide output voltages will be helpful to increase flexibility and compatibility with various LED characteristics. In this article, a PWM controller integrated with an advanced Primary-Side Regulation (PSR) technique will be introduced, and design guidance of the single-stage flyback converter suitable for wide output voltage ranges will be provided.

Primary Side regulation controller and its operation

Primary-Side Regulation (PSR) for LED drivers can be a solution for achieving international regulations (such as Energy Star®) for Solid-State Lighting (SSL) products. PSR controls the output current precisely with the information in the primary side of the power supply only, removing output current sensing loss and

eliminating secondary feedback circuitry. This makes it feasible to fit the driver circuit inside small-form factor retrofit lamps and meet international regulations without excessive cost increases for SSL application. Fairchild's FL7733 Pulse Width Modulation (PWM) PSR controller, helps simplify meeting SSL requirements while eliminating external components. The FL7733 provides highly precise output current regulation versus change in the transformer's magnetizing inductance, input and output voltage information, as well as powerful protection functions for system reliability.

Mode I

During the MOSFET turn-on time (t_{ON}), input voltage (V_{IN}) is applied across the transformer's primary-side inductance (L_m). Then, drain current (I_{DS}) of the MOSFET increases linearly from zero to the peak value ($I_{DS,PK}$), as shown in **Figure 1**. During this time, the energy is drawn from the input and stored in the inductor.

Mode II

When the MOSFET (Q) is turned off, the energy stored in the transformer forces the rectifier diode (D) to turn on. While

the diode is conducting, output voltage (V_{OUT}) and the diode forward-voltage drop (V_F), is applied across the transformer's secondary-side inductance and diode current (I_D) decreases linearly from the peak value ($I_{DS,PK} \times N_p/N_s$) to zero. At the end of inductor current discharge time (t_{DIS}), all energy stored in the transformer has been delivered to the output.

Mode III

When the diode current reaches zero, the transformer auxiliary winding voltage begins to oscillate by the resonance between the primary-side inductance (L_m) and the effective capacitor loaded across MOSFET (Q).

The output current can be estimated using the peak drain current and inductor current discharge time because output current is the same as the average of the diode current in steady state. The peak value of the drain current is determined by the CS peak voltage detector and the inductor current discharge time is sensed by the t_{DIS} detector. With peak drain current, inductor current discharging time, and operating switching period information; the innovative TRUECURRENT® calculation block estimates output current

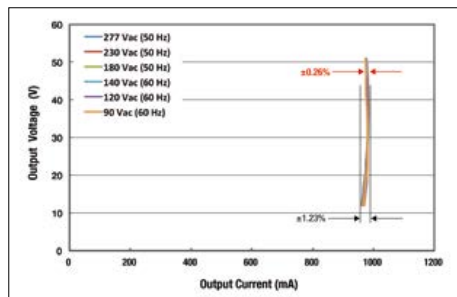


Figure 4. CC (constant current) performance over whole input and output voltage.

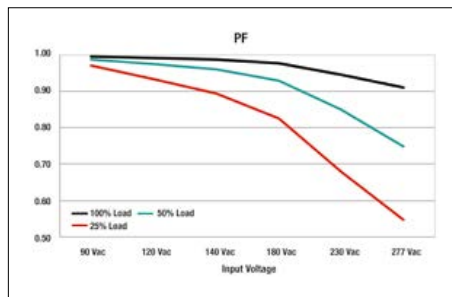


Figure 5. PF (power factor) performance versus input voltage according to load conditions.

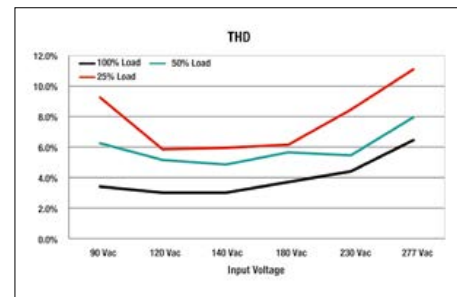


Figure 6. THD (transient harmonic distortion) performance versus input voltage according to load conditions.

According to Equation (3) the primary-to-secondary turn ratio is determined by the sensing resistor and output current as:

$$n_{PS} = \frac{I_0 \times R_S}{0.125} \quad (5)$$

$$n_{AS} = \frac{V_{DD.OVP}}{V_{O.OVP}} = \frac{23}{V_{O.OVP}} \quad (6)$$

$$n_{AP} = \frac{n_{AS}}{n_{PS}} \quad (7)$$

Since the saturation flux density decreases as temperature rises, the high-temperature characteristics must be considered if the transformer is used inside an enclosed case.

The first consideration for R1 and R2 selection is to set VS (voltage-sense) to 2.7 V to ensure high-frequency operation at the rated output power. The second consideration is VS blanking. The output voltage is detected by auxiliary winding and a resistive divider connected to the VS pin, as shown in **Figure 3**. However, in a single-stage fly-back converter without a DC link capacitor, auxiliary winding voltage cannot be clamped to reflected output voltage at low line voltage due to the small L_m current, which induces VS error. Frequency decreases rapidly at the zero-crossing point of line voltage, which can cause LED light flicker. To maintain constant frequency over the whole sinusoidal line voltage, VS blanking disables sampling at less than a particular line voltage $V_{IN.bnk}$ by sensing the auxiliary winding. The third consideration is VS level, which should be operated between 0.6 V and 3 V to avoid triggering SLP and VS OVP in wide output application.

Because the FL7733's V_{DD} operation range is 8.75~23 V, MOSFET switching will be shut down by triggering UVLO if output voltage is lower than $V_{OUT-UVLO}$ ($8.75 \times N_S / N_A$). Therefore, V_{DD} should be supplied properly without triggering UVLO across the wide output voltage range of 12~50 V. V_{DD} can be supplied by adding external winding N_E and V_{DD} circuits composed of a voltage regulator, as shown in Figure 3. The N_E should be designed so V_{DD} can be supplied without triggering UVLO at minimum output voltage ($V_{min.OUT}$). The external winding, N_E , can be determined as:

$$N_E > \frac{(8.75 + V_{CE.Q1} + V_{F.D3})}{(V_{F.D1} + V_{min.OUT})} \times N_S - N_A \quad (8)$$

where $V_{CE.Q1}$ is Q1's collector-emitter saturation voltage, and $V_{F.D3}$ is D3's forward voltage, and $V_{F.D1}$ is D1's forward voltage at the minimum output voltage.

$$R1 = n_{AP} \times \frac{V_{IN.bnk}}{I_{VS.bnk}} \quad (9)$$

where $V_{IN.bnk}$ is the line voltage level for VS blanking and $I_{VS.bnk}$ is the current level for VS blanking.

$$R1 = \frac{R2 \times 2.7}{(V_{OUT} + V_{F.D1})n_{as} - 2.7} \quad (10)$$

To show the validity of the design procedure presented in this application note, the converter described by the design example was built and tested. **Figure 4** shows the CC (constant-current) tolerance measured over the entire line and output voltage ranges. CC over universal line at the rated output voltage is less than $\pm 0.26\%$, and total CC regulation for whole line and output voltage range (12~51 V) is $\pm 1.23\%$.

Figures 5 and 6 show the PF and THD performance measured respectively at various load conditions.

Conclusion

The design considerations for a LED driver with wide output voltage as described here aim at increasing flexibility and compatibility for various LED lamp specifications and LED characteristics. Fairchild's PSR controller type FL7733 can provide good performance for high PF and low THD as well as constant current regulation over very wide output voltage range. In addition, the proposed LED driver can be used for a variety of LED lamps with simple design and low cost. ◀

(150370)

Inki Park – Application Engineer

Inki Park joined Fairchild Semiconductor in 2010 as an Application Engineer. He develops primary-side regulation controllers for LED drivers in addition to designing high-performance power converters with power factor correction.

Prior to Fairchild, Park spent six years developing a variety of power systems and LED drivers.

Park received a Master of Science and a Bachelor of Science in Electrical Engineering from Kangwon National University in Kangwon, South Korea.



LEARN

DESIGN

SHARE

elektor PCB Service

if offered in collaboration with

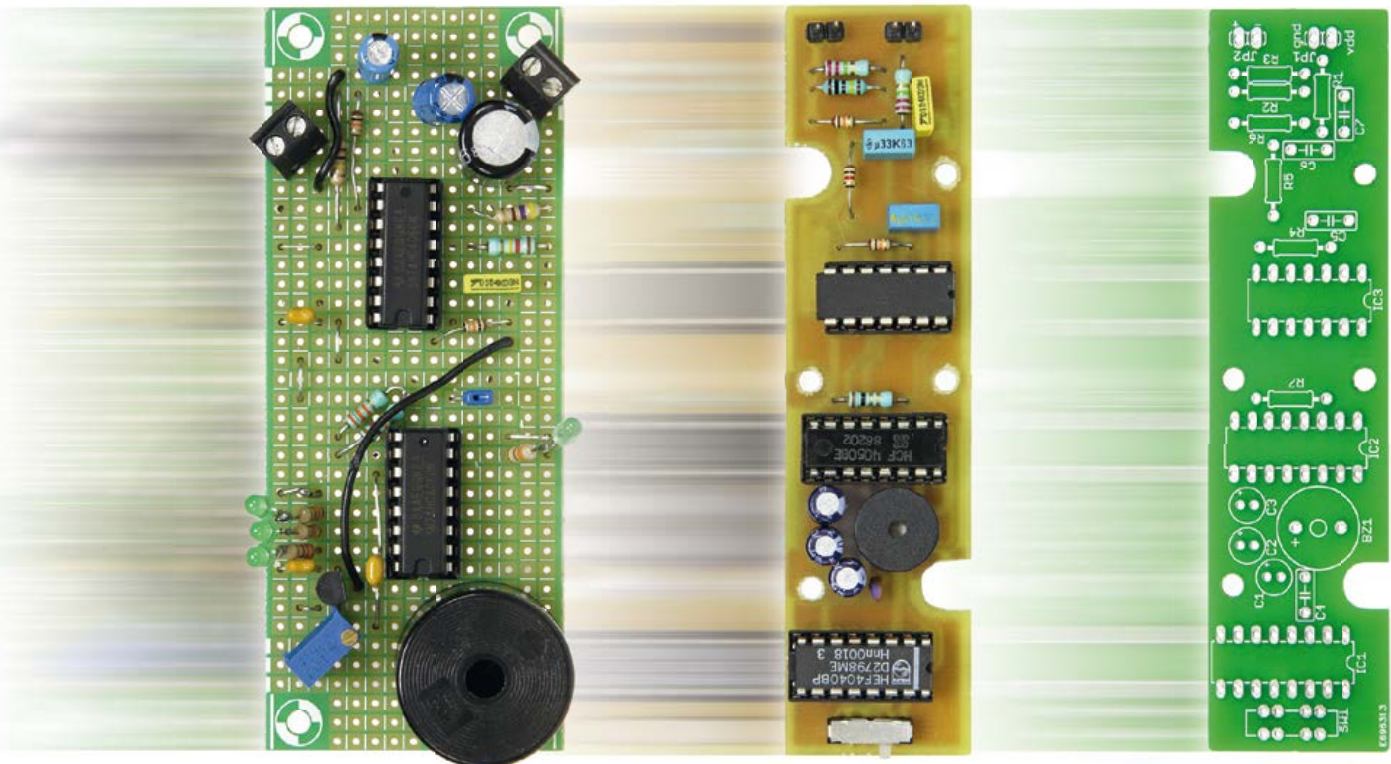


Generate your own PCB using the Elektor PCB Service

➔ Affordable

➔ High Quality

➔ Reliable



The Elektor PCB Service is the most extensive fully customized service for printed circuit board production in Europe. With convenient online tools allowing you to visualize and analyze your design before you order and pay .

- For beginners, there is the **NAKED-Prototype Service**:
This produces single and double-sided PCBs without solder masks.
- For a more advanced service, there is the **PCB Visualizer** that shows you how your PCB will look after production, with a **PCB Checker** performing a DRC for you and the **PCB Configurator** that lets you customize your order details.

Smart menus and select options guide you through the ordering process. You can see in advance exactly what our machines can produce so there won't be any surprises!



So start your next project here:

www.elektorPCBservice.com

Welcome to Elektor Labs

Elektor Labs is the place where projects large, small, analog, digital, new and old skool are sketched, built, discussed, debugged and fine-tuned for replication by you.

Our offer: Become Famous



Most engineers and budding authors we come across are just too modest. If they do not see the attraction and beauty of a design idea scribbled on a coaster and worked out later at home, others may, and should. Let Elektor Labs help you hone your design to perfection, let the editors assist with text & graphics, and reap the rewards by seeing your name in print in Elektor magazine's celebrated LABS section. Sure, we are happy to negotiate payment, but the actual remuneration will be fame & glory in electronics land, and your name added to the long list of extremely successful e-authors. Our get-u-famous formula also applies to book authors, bloggers and video makers. Students and youngsters: being in publication is a current boost like no other in getting a job!

Our Experts and Designers

Besides experienced support staff and BSc, MSc qualified engineers with a total working life in electronics of about 200 years, the Lab has access to Elektor's vast network of experts for consultation, critical advice, and assistance with specialized assignments.

Our Facilities

We are sumptuously housed in three spacious rooms at Elektor House where we try unsuccessfully to keep our solder jobs and prototypes off our computer desks. We have water, 218 volts electricity and coffee nearby. PCB milling, prototype assembly, SMD reworking, audio testing, pizza cooking, and mechanical work are deferred to converted cellars in the basement.

Our Standards

All projects and products going through the Labs pipeline are produced to high engineering standards. In practice, prototypes of projects labeled LABS in the magazine must be demonstrated to work to specification on certified, calibrated test equipment available locally. BOMs and schematics must match perfectly. Kits are sampled for completeness. We are ROHS compatible, lead free and comply with electrical safety standards applicable in our location. If engineering errors are found these will be put up for public notification.

412

Project Proposals

65

Projects in Progress

182

Projects Finished

683

Projects Total

Our Products

Our products are visible in Elektor magazine, as well as on the Elektor.Labs and Elektor Store websites. The range includes text write-ups for editors, prototype photography, PCBs including SMD-prestuffed, PCB files, project software, programmed devices, semi-kits, tools, modules, videos, and service desk information.

Our Webinars

The more talkative of our Lab engineers do not stop at testing prototypes, they happily share technology related problems, insights, get-u-going information, and design skills on live camera at Elektot.tv. Labs' webinars are free to attend and extremely interactive. They are announced in Elektor.POST, and webcast live from Elektor House in Holland. Do plug in!

Our History

The origins of Elektor Labs go back to the early 1970s when soldering and writing was a one-man, single-desk job. Over the years Labs staff have not only witnessed the arrival of the transistor, the IC, the microcontroller, and the SMD, but actually jumped on these parts as soon as they were out of the professional-only woods. Once special branches, Audio Labs, Micro Labs, PCB Labs, and Mechanical have converged back again into a single activity.

elektor labs

Sharing Electronics Projects

Home Proposals In Progress Finished

Upload your own projects!

On our very own Elektor-Labs website, you can share and showcase your ideas and project proposals to thousands of other electronic engineers. The site also allows you to follow other specialists' activities, supply comment, and so push the projects forward. Here's the best part, the top projects are eligible for processing in our test lab, in preparation for publication in Elektor magazine.

Who's this for?

Although only members can log on to our Elektor.Labs website to publish projects and contributions, anyone can view along with the other projects. If you'd like to see your project published in Elektor magazine, which is published in four languages and read by tens of thousands of electronics specialists all over the world, then become a Green of Gold member (www.elektor.com/member) or log in as an existing member of our Elektor community!



Welcome to the **DESIGN** section

By **Clemens Valens**, Elektor Labs

SHARE

DESIGN

LEARN



Strive for the best

The other day I took a flight from Paris to Mumbai. Just before boarding at Paris Charles de Gaulle (CDG), at the gate, I noticed a sign saying that CDG was competing for the title of "World's Most Improved Airport 2015". Odd, I thought. First of all because I use CDG several times a year and I hadn't noticed any improvements at all, my flights still are at one hour or more from the entrance as well as delayed – and believe my surprise when at the gate instead of a plane I found a city bus labelled "Mumbai" – but mostly because of the ambition, or more precisely, the lack of it that emanates from the award.

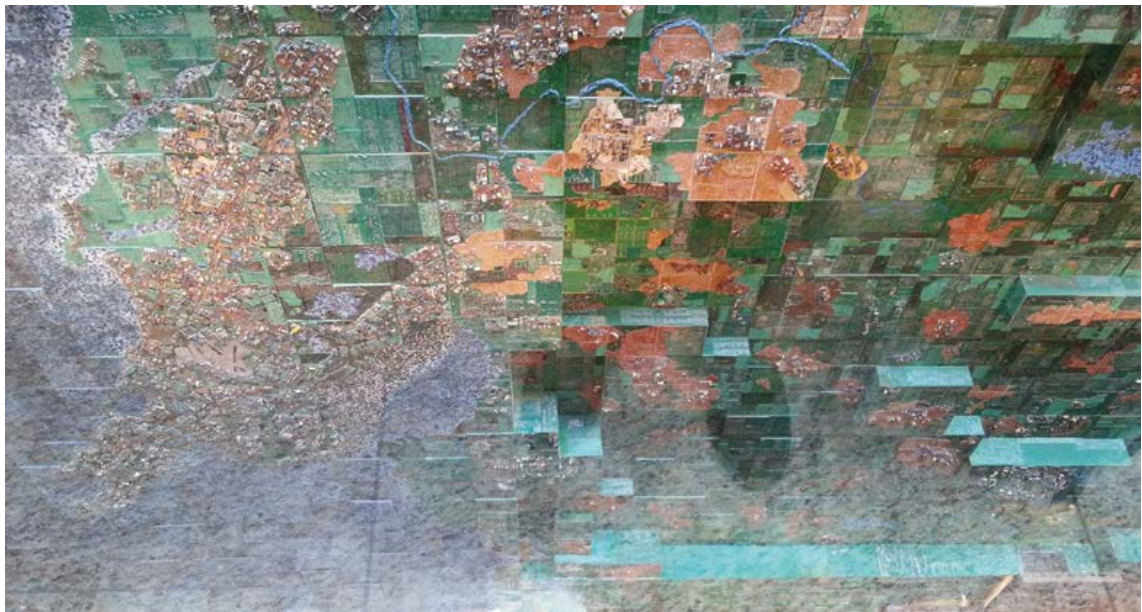
Imagine you design a product, a computer operating system for example, that's really awful but you commercialize it anyway. Then you change it somehow and make it win the "World's Most Improved Product" award. Does this mean that the product is now a good product? No, it can still be lousy, and it probably is, but less lousy than before. The award is totally void of any meaning. To come up with a good product you must strive for the title of "World's Best Product". Only when you go for the best can you come up with a quality product. If only more com-

panies would do that the world would be a better place.

P.S. Did I mention that the 2015 World Airport Awards event was held in Paris? Needless to say that CDG won the award.

www.worldairportawards.com

The art of electronics



Mumbai Chhatrapati Shivaji International Airport (BOM) came in third in the World's Most Improved Airport 2015 elections. I was there before and after this airport got rebuilt and I can only applaud this recognition. Actually a higher (first?) position would have been justified as it now is a very nice and modern airport where liquids pose no problems in security checks.

Strolling along the quiet corridors decorated with works of art from all over India I noticed a huge mosaic representing a map of Mumbai and its backlands. Looking closer I discovered that it was entirely made of circuit boards, electronic components and wires. Imagine an ocean of aluminium capacitors, rivers of wire and fields of populated FR4. If you ever get the chance to go there, have a look, it is at gate 85/86.

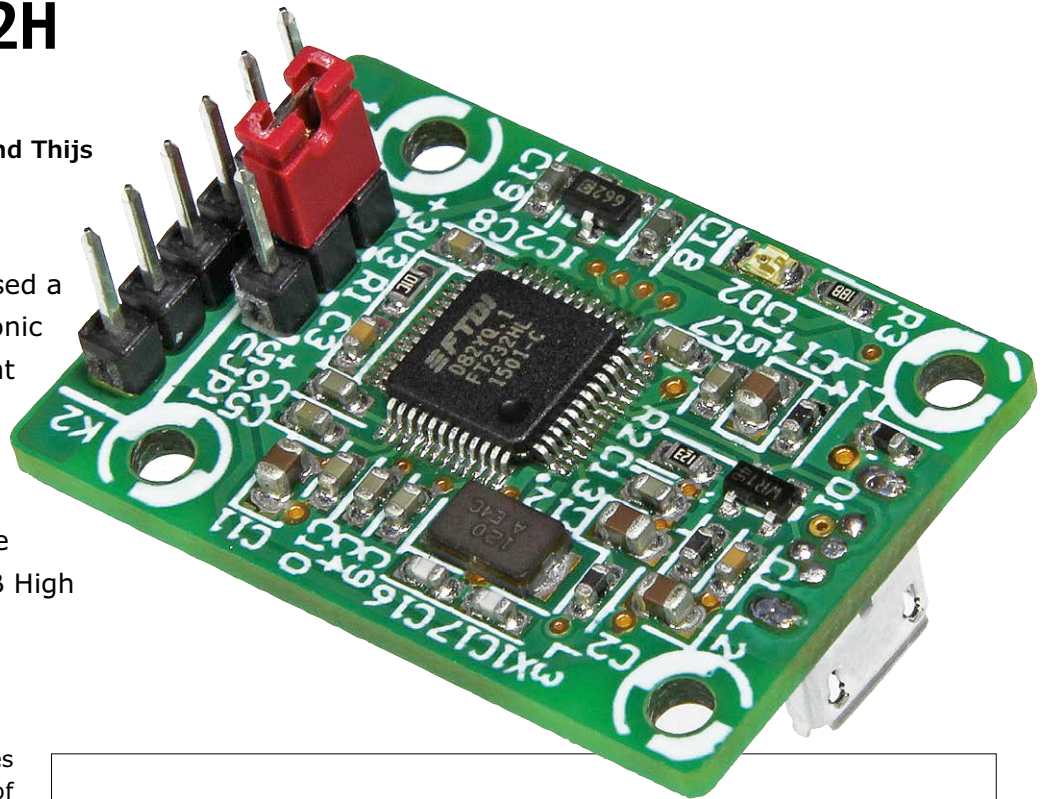
The photo is a bit hazy because of the bullet-proof glass. ◀

(150476-I)

Compact USB to Serial Converter Using the FT232H

By Ton Giesberts (Elektor Labs) and Thijs Beckers (Netherlands Editorial)

The “legacy” serial port is still used a lot for interfaces between electronic devices and PCs. Here we present a compact implementation, based on the FT232H IC from FTDI, which can be connected over USB 2.0 after you install the right drivers. That gives you USB High Speed capability.



What? Another serial port converter? Yes indeed — we just can't get enough of them. We recently presented a USB to multi-protocol serial converter based on the FT232H from FTDI (Elektor May/June 2015, page 90 [1]). The project described here is an extension of that design and is intended for panel mounting. For that reason, the USB connector is mounted on the rear side. That makes it very handy for quickly adding USB capability to a project, but it prevents the converter from being used as a breakout board. Here the main aim was to keep things small, which we managed reasonably well with dimensions of 22 × 33 mm.

Features

- USB 2.0 High Speed
- Pin compatible with the FTDI-USB-TTL (UART) cable
- Right-angle USB connector for space-saving mounting
- Selectable 3.3 V or 5 V supply voltage on the output connector
- Automatically recognized as a COM port (after installation of VCP drivers [3])

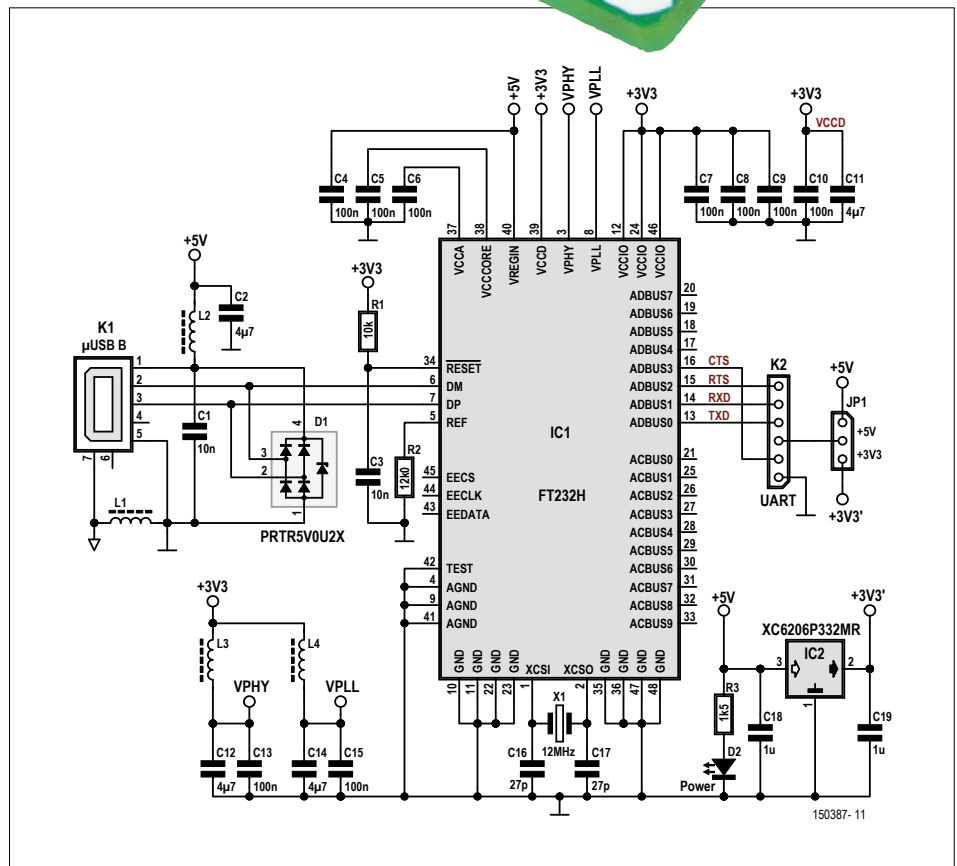


Figure 1. The schematic is a slimmed-down version of the USB to multi-protocol serial converter published in Elektor May 2015.

MPSSE

The Multi-Protocol Synchronous Serial Engine (MPSSE) is a flexible interface between synchronous serial devices and a USB port. Three lines are necessary for a serial link: data, clock and ground. The clock is usually generated by the master device. The other devices on the bus synchronize their data transmission and reception to this clock. Some examples are the SPI, I²C and JTAG buses. The UART of a microcontroller serial port is an example of an asynchronous serial interface. Here the data and clock signals are combined and the clock must be reconstructed from the data signal.

The MPSSE supports communication with various types of serial device. The data formatting and clock synchronization can be configured in different ways, with data rates up to 30 MB/s. Parallel communication is also possible, for example in FT1248 mode. With this form of half-duplex communication, you can trade off bandwidth against speed with 1, 2, 4 or 8 data lines. Synchronous and asynchronous FIFO modes, such as FT245, are also possible. And of course, you can do bit-banging if you want to.

Full Speed or High Speed?

Although the USB 3.0 standard was released a fair while ago and its successors — USB 3.1 and more recently USB Type C — can already be found on the latest generation of laptop and desktop PCs, USB 2.0 speed is more than adequate for RS-232 and RS-485 applications or communication with the serial port of a microcontroller. The FT232H meets the USB 2.0 High Speed specification (480 Mbit/s), and its I²C, SPI and parallel port capabilities make it more versatile than the FT232R, which only meets the Full Speed specification (12 Mbit/s). The extra capabilities of the H version are implemented in the Multi-Protocol Synchronous Serial Engine (MPSSE). This is explained in detail in the **MPSSE** inset. However, these capabilities are not used in this project. If you need them, please see the USB to Multi-Protocol Serial Converter article [1]. In any case, it's more logical to use the H version, and it costs less (at the time of writing) than the R version.

Component List

Resistors

0.1W, 1%, SMD 0603
R1 = 10k Ω
R2 = 12k Ω
R3 = 1.5k Ω

Capacitors

Default: SMD 0603
C1,C3 = 10nF, 50V, 10%, X7R
C2,C11,C12,C14 = 4.7 μ F, 10V, 10%, X7R, SMD 0805
C4-C10,C13,C15 = 100nF, 50V, 10%, X7R
C16,C17 = 27pF, 50V, 5%, COG/NP0
C18,C19 = 1 μ F, 6.3V, 20%, X5R

Inductors

L1-L4 = 600 Ω @ 100MHz, 25 %, 0.15 Ω , 1.3A, SMD 0603

Semiconductors

D1 = PRTR5V0U2X (SMD SOT-143B)
D2 = LED, green (SMD 0805)
IC1 = FT232HL (SMD LQFP-48)
IC2 = XC6206P332MR (SMD SOT-23)

Miscellaneous

K1 = Micro-USB 2.0 type-B through-hole PCB connector, vertical
K2 = 6-pin SIL socket, 0.1" pitch
JP1 = 3-pin pinheader, 0.1" pitch
JP1 = jumper, 0.1" pitch
X1 = 12MHz quartz crystal, C_L = 18pF, SMD 5 x 3.2mm
PCB # 150387-1

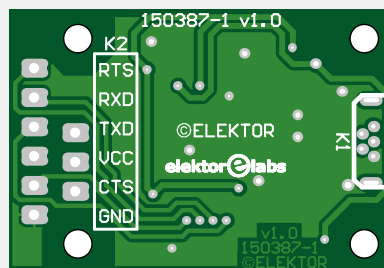
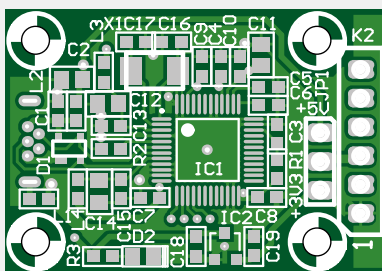
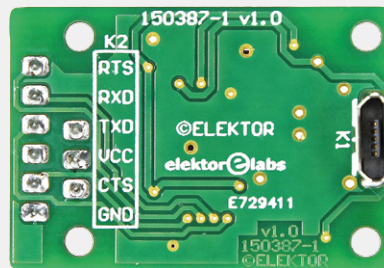
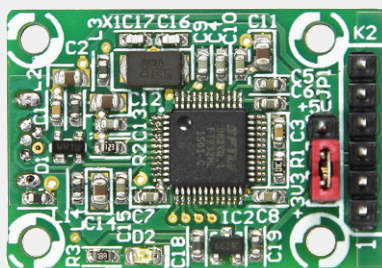


Figure 2. Virtually every square millimeter is used on the miniaturized PCB.



▶ It's nice to have that High Speed feeling

Find the differences

If you look at the schematic diagram in **Figure 1**, you're bound to see a lot of similarities to the original design. The FT232H is supported by a 3.3 V voltage regulator (IC2) and the usual filter and decoupling capacitors. A special protection diode guards the inputs of the FT232H against static discharges up to 8 kV. The 3.3 V supply voltage is connected to the $\overline{\text{RESET}}$ pin through R1/C3 as specified by the data sheet. Another thing specified by the data sheet is resistor R2 (12 k Ω , 1%), which ties the current reference input REF to ground. The 12 MHz crystal (tolerance ± 30 ppm) is also a mandatory component, along with the tuning capacitors C16 and C17. L1, L2 and C1 provide a bit of protection against high-frequency noise on the supply line and the shield. The power indicator LED (D2) survived the slimming-down exercise and is handy for seeing whether the module is powered up.

K1 is a Micro USB connector, which is

Table 1. Connector K2 pinout (FTDI cable)

Pin	Function	Color
1	GND	Black
2	CTS (AD3)	Brown
3	VCC	Red
4	TX (AD0)	Orange
5	RX (AD1)	Yellow
6	RTS (AD2)	Green

mounted on the back of the PCB. The serial UART signals are routed to the FT232H through K2. Here the pin layout has been kept compatible with the FTDI cable [2]. **Table 1** shows the signal assignments. Jumper JP1 selects either 5 V or 3.3 V as the output voltage on this connector (see **Table 2**). Just as in the original design, voltage regulator IC2 is connected directly to the 3.3 V line to provide the 3.3 V supply voltage for the

Table 2. Jumper JP1 settings

Connection	Function
1-2	K2 pin 3 is 5 V
2-3	K2 pin 3 is 3,3 V

FT232H.

The EEPROM that enables configuration of the USB to multi-protocol serial converter has been eliminated to cut cost and save space. It is thus not possible to configure the FT232H in different modes with FT Prog, so it only works in the default UART mode.

Practical aspects

The small PCB (see **Figure 2**) has four mounting holes for securing it with M2 screws. The USB connector is mounted on the rear of the board, and we choose a right-angle type for this model so it can be accessed through a relatively thick front or rear panel.

When it is connected to a PC, the FT232H is immediately recognized as a UART if the Virtual Com Port (VCP) drivers [3] are already installed. **Figure 3** shows how the FT232H is recognized in Windows 7. If it doesn't work for you, there are lots of descriptions on the Internet that can help you, such as on the Adafruit website [4]. ◀

(150387)

Cypress CY7C65211

FTDI is not the only manufacturer of ICs for serial interfaces. For example, Cypress offers the CY7C65211, which in their words is a "USB-Serial Single-Channel (UART/ I²C/SPI) Bridge with CapSense® and BCD". This IC has a number of attractive features, including an integrated EEPROM (eliminating the need for an external IC), bus mode selection through the API (with FTDI you have to use a configuration tool), and support for serial ports with signal levels from 2.2 to 5 V.

Nevertheless, we decided to stay with the FT232H because it supports the USB 2.0 spec. It's nice to have that High Speed feeling, and the FT232H is "factory configured" in UART mode, so you can use it right out of the box.

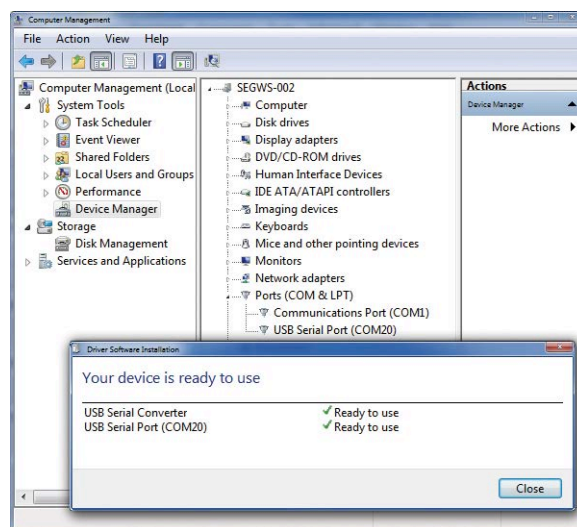
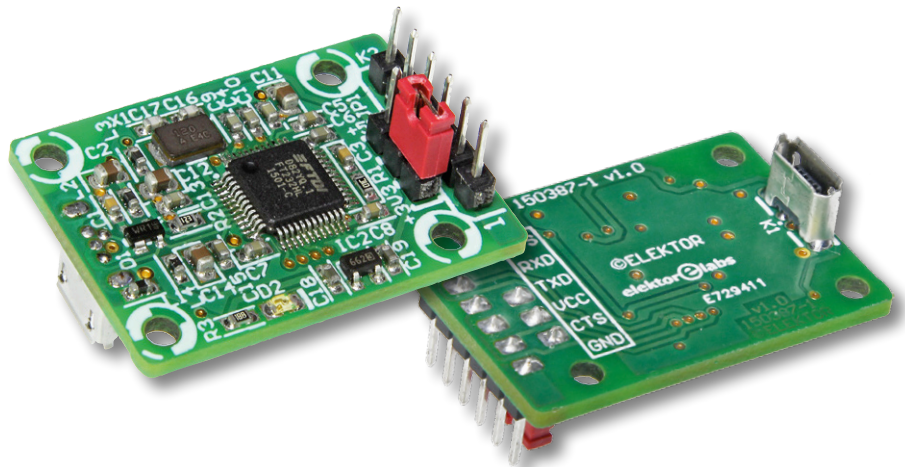


Figure 3. After the VCP drivers are installed on the PC, the FT232H is recognized immediately as a COM port.

Web Links

- [1] USB to Multi-Protocol Serial Converter, Elektor May/June 2015: www.elektormagazine.com/130542
- [2] FTDI cable: # 080213-71 (5 V version) and # 080213-72 (3.3 V version) at www.elektor.com
- [3] FTDI driver downloads: www.ftdichip.com
- [4] Adafruit installation help: <https://learn.adafruit.com/adafruit-ft232h-breakout>

Android I/O Board (2)

Control embedded electronics from your Android phone or tablet



By **Elbert Jan van Veldhuizen** (The Netherlands)

In this installment the Bluetooth, WiFi, USB and USB-Host modules are described. We'll also show you how the Android I/O board can be controlled via an app. As an example, we're going to create an app for the exposing and etching of circuit boards.

Modules

We start with descriptions of the specifications and features of the various modules that can be connected to the I/O board.

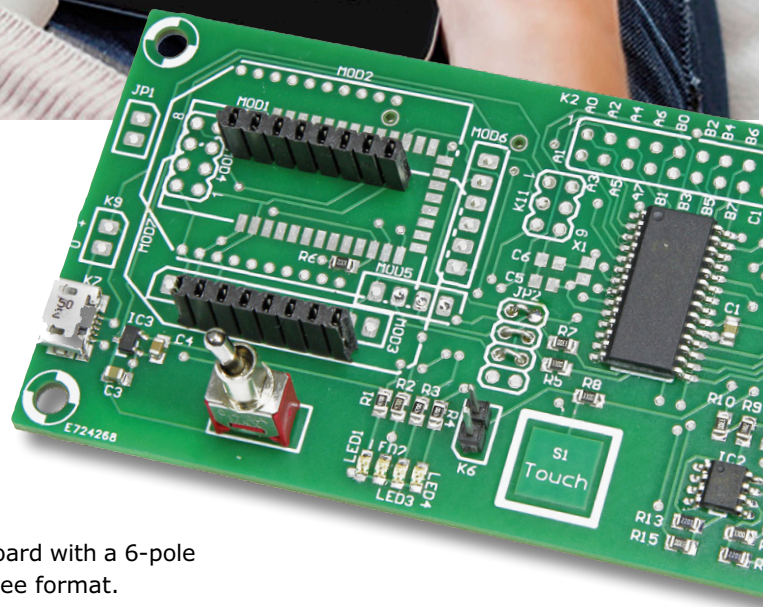
HC-06 Bluetooth

The HC-06 is an easily obtainable Bluetooth module. This is available as a break-

out module, a circuit board with a 6-pole connector, or in a ZigBee format.

All the data received by the HC-06 via its Bluetooth connection is transparently forwarded to the serial port. The default setting for the serial port is 9600 baud, 8 bits, no parity and 1 stop bit. You have to select the 'HC-06' device in the Bluetooth

settings when you want to make a connection to the module with your Android device (see **Figure 1**). They then have to be paired, for which the pin code is 1234. In the pairing menu you will see a mention of its Bluetooth address (xx:xx-



:xx:xx:xx:xx). Make a note of this, as it will be needed in the app.

When a connection has been made, the green LED is on continuously. When there is no connection, this LED will flash. The HC-06 can support only a single connection and operates as a slave; this means that the Android device has to initiate the connection.

RN-41/42

According to the datasheets, the RN-41XVC (high-power Bluetooth) and the RN-42XVC (low-power Bluetooth) will also fit on the board. The modules (note the 'XVC' in the part number) have a Zig-Bee footprint. However, the author has not tested these modules. If any readers try these out, by all means let us know how you get on.

Elektor BL600 E-BoB

This Bluetooth 4.0 module can also be mounted on the board. Bear in mind that you have to set it up in advance to operate in its transparent mode (BT4 <-> UART).

RN-171XV

The RN-171 is a WiFi module made by Microchip. This module transparently forwards all data received over the TCP/IP connection (established via WiFi) to the serial port. The protocol is (as with the HC-06) 9600 baud, 8 bits, no parity and 1 stop bit. This module is more complicated than the HC-06 in two aspects. First of all, the module needs to make a connection to the WiFi network. But you need

a working connection before you can set the parameters (such as the SSID and password) for the network. This is very much a chicken and egg problem. The easiest way round this problem is to temporarily change the SSID and password of your WiFi network to that of the standard settings of the RN-171, which are 'roving1' and 'rubygirl'. You can then use a terminal program such as TeraTerm Pro (see part 3) to log on to the RN-171. The green LED shows the state of the connection. When it flashes very quickly, it means that the module is not logged on to any WiFi router. When it flashes slowly, it means that the module has logged on to the WiFi router. It's only when the green LED is on

continuously that a TCP/IP connection has been established. The yellow LED flashes whenever data is received or transmitted.

There are several other ways in which you can make a connection. Instead of changing the settings of the WiFi router, you could set the Android device to tethering mode (using the SSID and password mentioned earlier). The Android device now functions as a router and you can use a telnet app (for example, 'telnet' is available from the Play Store) to make a connection. The RN-171 can also create an ad-hoc network; this type of network is supported by Windows, but not by Android. Lastly, you can make a connection directly via the serial port of the RN-171 module (using a USB-serial cable, for example). This is done via the SERIAL jumper on the board. Remember to disconnect it from the microcontroller first.

The RN-171 can operate in two modes. By default, the module forwards all data received over the TCP/IP connection to the serial port. However, when it comes across the character string \$\$\$, the module switches to its interactive mode. You will then have access to a number of commands to configure the RN-171. You're now also able to directly control some of the ports on the module [1], but we won't make use of that in this instance. **Table 1** has a list of the commands needed to configure the module.

The RN-171 can only handle a single TCP/IP connection at a time. The RN-171 uses port 2000 by default. The RN-171 can also initiate a TCP/IP connection itself, but we won't make use of this in this project.

Table 1. Commands for configuring the RN-171 module

Command	Remarks
\$\$\$	Enter interactive mode
scan	Scan for WiFi networks (not required if the SSID is known)
set wlan ssid <SSID>	Set the SSID of the router (or access point, AP)
set wlan passphrase <password>	Set the password of the AP
set wlan join 1	Attempt to make a connection with the AP
save	Save the settings in the memory
join SSID	Log on to the network
exit	Leave the interactive mode and start a transparent connection

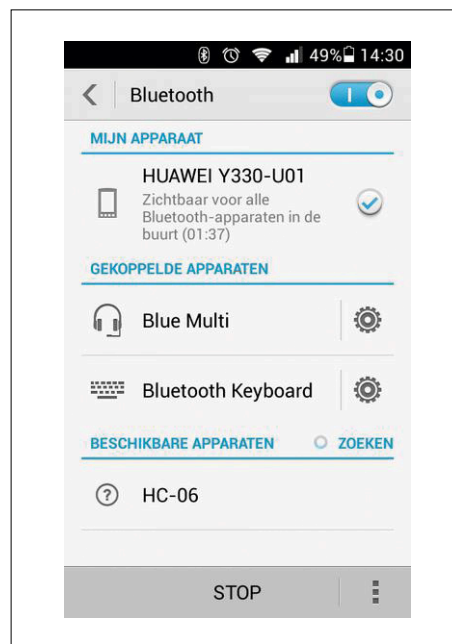


Figure 1. The Bluetooth pairing menu.

ESP8266

The ESP8266 WiFi module has only recently come onto the market, and is very inexpensive. This module is available in several formats, although the most popular has 8 (2x4) pins. This module can be mounted directly onto our board. The firmware of the ESP8266 module can be updated using jumper JP1 on the board. However, the firmware is still under development at the time of writing. For example, the baud rate depends on the version of the firmware (115,200 baud for the author) and this can't be changed. The module can be set up as an Access Point (where the Android device logs on to the module), or it can log on to an

Access-Point. The latter mode is not yet fully implemented. Furthermore, the data is not forwarded transparently, but it has to be requested via AT commands. This means that the firmware of the Android I/O board has to be modified. It is the intention that this module will be covered in a future article once a stable version of the firmware is available.

Elektor Android Break-out Board (# 130516-91)

This Break-out Board, which uses an FT311D, was described in the June 2014 edition of Elektor Magazine. This board fits in the 10-way connector (MOD3) on the Android I/O board. There are also other modules available that use the FT311D. Since then, FTDI has released the FT312D. Whereas the FT311D can operate using several protocols (including the serial connection), the FT312D has just the serial connection. This is therefore suitable for use with the Android I/O board.

You need Android version 3.2 or later in order to use the USB Accessory-mode (and therefore to use these modules). In practice, it turned out that the module didn't work with a number of Android devices, despite having the correct Android version.

The Elektor Android Break-out Board supplies both 5 V as well as 3.3 V. For small currents you can use the 3.3 V supply directly. All you need to do is to connect the top two pins of JP2 together and remove IC3 (the 3.3V voltage regulator on the Android I/O Board).

If you expect to need a larger current, you should use the 5 V supply. In that case, IC3 should be used and the top two pins of JP2 should not be connected together. The serial connection of the FT31xD chips requires a longer stop bit than usual. Even when it is configured for 1 stop bit, the FT31xD expects a longer stop bit. If this is not taken into account, you'll find that you will receive strange characters in the Android app. The author has therefore modified the firmware, increasing this period. This firmware is identified by the 'FT31xD' in its file name.

It should also be noted that the FT31xD must not receive characters from the Android I/O board before the FT31xD has been enumerated. In practice, this means that the Android board should not automatically send the results, such as the time, ADC values, or changes in the

CPS and input states. This default behavior can be configured from the app.

The best chance of success is achieved when you first connect the Break-out Board (with the FT31xD) to the Android device, before starting the app. Once Android has recognized the connection, you can start the app.

Elektor FT232R USB/Serial Bridge BoB (# 110553-91)

It is also possible to directly mount an Elektor FT232R USB/serial breakout board on the Android I/O board. Other modules based on the FT232x can also be connected, if necessary via discrete wires. In any case, these modules work with PCs using the drivers from FTDI. On Android devices you'll need a USB-host port or a USB OTG port. Most tablets will have one of these (even the cheaper ones), but for smartphones this is less likely. In addition, the Android device has to be set to 'USB debugging' mode (under settings/developer options/USB debugging).

The Elektor FT232R USB/serial BoB can be configured to supply 3.3 V or 5 V. In contrast to the FT311D board, only one voltage is output in this case. The PIC16F1938 microcontroller and the I²C flash memory can operate at either voltage. You can then choose to make the Android I/O board to operate with a 5 V supply, if this is advantageous for the I/O. With a 5 V supply you should adjust the values of the current limiting resistors (R1 to R4) for the LEDs. When the PIC16LF1938 is used, the supply can only be 3.3 V. In either case, IC3 should not be mounted and the top two pins of JP2 should be connected together.

Board connector JP2

Board connector JP2 takes care of the connections for the serial data and the supply for the modules. During normal operation, the top three pairs of pins have to be connected together. JP2 can also be used to test the microcontroller and the communications module independently. You can connect a USB/serial converter to JP2 to communicate directly with the module or microcontroller, using a terminal program on a PC. The left side of the connector goes to the microcontroller, the right side to the module. Note that you can't use the same cable for both sides, since the TX and RX connections have been swapped over on each side (for the jumpers).

The author used the program TeraTerm Pro [2] and the Elektor FT232R USB/serial BoB. For TeraTerm Pro the serial connection appears as a COM port. In setup/serial port you can set the baud rate. In setup/terminal you should set newline to 'auto' and 'CR'.

Standard classes

To make it easier to program your own app for the Android I/O board, the author has written a number of standard classes to set up the Bluetooth, WiFi and USB connections. The classes are called **BTFunctions**, **WiFiFunctions**, **USB-Functions** and **USBHostFunctions** (from here on called xFunctions). The classes have the following five methods:

- `void initiate(int channel, handler messageHandler <,String address> <, int port>, context)`
- `void connect()`
- `void disconnect()`
- `boolean connected()`
- `void send_command(String command)`

In *initiate* you have to pass a handler and the address of the Android I/O board. The handler is the communications channel between xFunctions and the app. There are two 'sub-channels'; One channel is used to communicate the state of the connection, the other channel passes on all the data that is sent back from the Android board. A message handler in the app has to receive and process this data. The address for Bluetooth is the set of 6 hexadecimal numbers (separated by colons); for WiFi it is the IP-address and port number, and for USBHostFunctions the port number is the board number, where '0' is the first board, '1' the second, etc.

The (main)channel is used to distinguish between the boards when you connect more than one Android I/O board. Each connection will now have a unique number, which is sent along to the message handler. This way you can differentiate between the various data streams. When the app supports a single board, the value of this number doesn't matter. The context can be obtained using 'this'.

With *connect* you create the actual connection, and *disconnect* breaks the connection. It is recommended that you break the connection whenever the app

is 'out of view', for example when the user clicks on the home button, or when another app appears on top and gets the focus. If we didn't do this, the connection would remain active. With a Bluetooth connection the Android device would continue to use more power. Another problem would be that it would not be possible for another app or user to log on to the Android I/O board, since the modules can only have a single connection at any time. Android calls *onPause()* when the app loses focus; from here we should also call *disconnect*. We have to call *connect* from *onResume()*, which is called by Android when the app becomes visible (both when it first opens and when it regains the focus).

The transmission of data happens with *send_command*. The string is sent exactly as it is in the parameter, without any extra carriage returns or a leading '0'-byte'. The reply from the Android I/O board is passed on to the message handler.

The correct 'permissions' have to be set in the AndroidManifest.xml file to gain access to the Bluetooth, WiFi and USB ports. The user will see these when the app is installed. The permissions are:

WiFi:

- `<uses-permission android:name="android.permission.INTERNET" />`
- `<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />`

Bluetooth:

- `<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />`
- `<uses-permission android:name="android.permission.BLUETOOTH" />`

USB Accessory:

- `<uses-feature android:name="android.hardware.usb.accessory" />`

USB Host:

- `<uses-feature android:name="android.hardware.usb.host" />`

One area that caused particular problems for the author was with the programming

Using an NTC to calculate the temperature

An NTC is a resistor with a negative temperature coefficient. The resistor value quoted by the manufacturer is valid at 25 °C (i.e. room temperature), and decreases as the temperature increases. The resistor value changes according to a logarithmic scale, where the curve is determined by a constant called the beta value. The value of this is usually between 3000 K and 4000 K, and is stated in the datasheet by the manufacturer. The temperature (in degrees Celsius) can be calculated using the following formula:

$$T = \text{beta} / \ln(R_{\text{NTC}} / R_{00}) - 273.15$$

where $R_{00} = R_{\text{NTC}@25^\circ} \times \exp(-\text{beta}/298.15)$

When the NTC is part of a potential divider, where the resistor is in the positive branch and has the same value as the NTC at 25 °C, the temperature is calculated as follows:

$$T = \text{beta} / \ln(1 / (\exp(-\text{beta}/298.15) \times ((V_{\text{ADCmax}} / V_{\text{ADC}}) - 1))) - 273.15$$

where V_{ADC} is the measured value and V_{ADCmax} is the voltage across the potential divider.

of the Bluetooth functions. Not all Android devices have implemented the Bluetooth API properly, with the API not always giving the correct reply when a connection is made or broken. Some extra checks were therefore added to this class to make certain that the connection was actually made.

If you want to write your own program for the USB-Host feature you have to download the d2xx.jar library from the FTDI website [3] and save it in the libs directory. You don't have to do anything extra to install the APK, since the jar file will be automatically included in the apk file. More details on programming your own apps using these classes can be found at the start of the source files.

Class IOBoardFunctions

If you want to give the app the ability to communicate via Bluetooth, WiFi as well as USB (for example, via a selection from a menu in the settings), you can use the wrapper *IOBoardFunctions* instead of the classes mentioned above. This has the same methods as the individual functions. The only difference is in the method *initiate*, which requires an extra parameter that indicates which communications method should be used. The values 0, 1, 2 and 3 select Bluetooth, WiFi, USB Accessory and USB Host respectively. The class remembers this and ensures that the correct communications channel is used in all other functions. This makes the programming much easier.

Android app: Etching Controller

An example app has been created, which can be downloaded from the ElektorMagazine website [4], and which includes the source code. The app helps you expose and etch your own printed circuit boards. A timer controls the light box. There are two thermostats for the temperature control of the developing and etching tanks. Each tank has its own timer that gives a signal (via a buzzer and an alarm on the Android device itself) when it's time to remove the board from the tank.

To install the app, you have to copy the apk file from the bin directory (of the download) to the flash memory of the Android device. This apk file is then installed using the file manager or apk-installer. In 'Device Administration', under 'Security', you should first tick 'Unknown sources - Allow installation of apps from unknown sources'.

If you want to examine this app and modify it, you'll have to install an Integrated Development Environment (IDE) on your PC. Eclipse is an IDE for the development of java applications and the Android SDK can be used to program and debug Android apps. You can download Eclipse and the Android SDK as one package from the Android website [5]. Another option is to use Android Studio, which can be downloaded from [6].

Download the source files and unzip them in a working directory. When you start

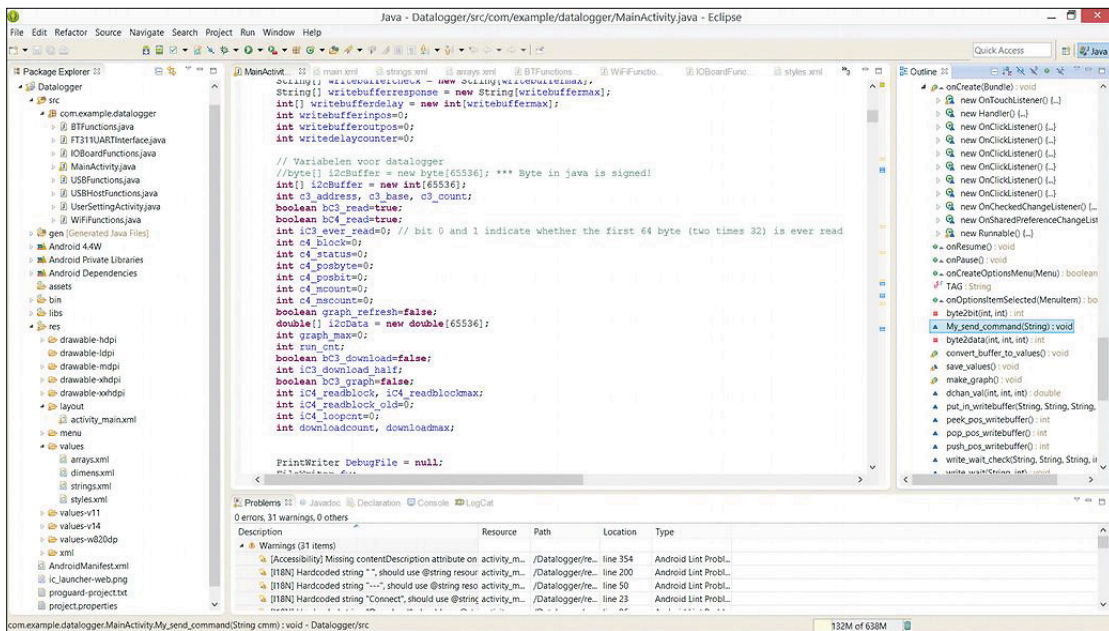


Figure 2. The screen layout of Eclipse, with the project manager (left), source editor (top, center), outline view (right), and the consoles (bottom). Android Studio has a similar layout.

Eclipse it will ask for a workspace; it's best to use a new, empty directory for this. Once you've started Eclipse you can use 'File/Import.../Android/Existing Code Into Workspace' to import the code. Make sure that you tick 'Copy projects into workspace'.

By default, Eclipse shows an overview of the files in the project on the left (see **Figure 2**). In the middle is the source file being edited. For some files you will see different tabs at the bottom, which let you edit the file in different ways. For

example, one option with layout files is to edit them graphically. Error messages are (in text mode) shown in the left and right of the margin. When you click on the symbol in the left margin you will be presented with possible solutions, which can be included in the code with a double-click. On the left is the structure of the file. For a java file we will see the functions and variables, along with their interdependencies. At the bottom are various consoles, such as an overview of all errors and warnings, the compilation status of the project and debug information.

A project can be compiled via the menu: 'run/run configurations', or by clicking on the icon in the taskbar.

An overview of the files needed for our example app is shown in **Table 2**.

The overall structure of the app is shown in **Figure 3**. The main code is in the file `src/com.example.EtchControl/MainActivity.java`. When the app starts, Android calls `onCreate()`. This is where everything has to be initialized.

Sending data to the Bluetooth port (for example) and reading from it are done in two individual streams that are com-

Table 2. Important files used in programming the example app

File	Function
<code>src/com.example.EtchControl/MainActivity.java</code>	The main code for the app
<code>src/com.example.EtchControl/BTFunctions.java</code>	The class for the Bluetooth connection to the Android I/O board
<code>src/com.example.EtchControl/WiFiFunctions.java</code>	The class for the WiFi connection to the Android I/O board
<code>src/com.example.EtchControl/USBFunctions.java</code>	The class for the USB-Accessory connection to the Android I/O board
<code>src/com.example.EtchControl/USBHostFunctions.java</code>	The class for the USB-Host connection to the Android I/O board
<code>src/com.example.EtchControl/IOBoardFunctions.java</code>	The class as wrapper for the connections to the Android I/O board
<code>src/com.example.EtchControl/UserSettingActivity.java</code>	The (standard) code for the settings menu
<code>res/layout/layout.xml</code>	The layout of the GUI in portrait mode
<code>res/layout-land/layout.xml</code>	The layout of the GUI in landscape mode
<code>res/xml/user_settings.xml</code>	The structure of the settings menu
<code>res/menu/menu.xml</code>	The structure of the menu
<code>res/values/strings.xml</code>	The text strings for the GUI, the settings and the menu
<code>res/values/arrays.xml</code>	The arrays for all the options in the settings
<code>AndroidManifest.xml</code>	The permissions for the app
<code>bin/Etchcontrol.apk</code>	The compiled apk file

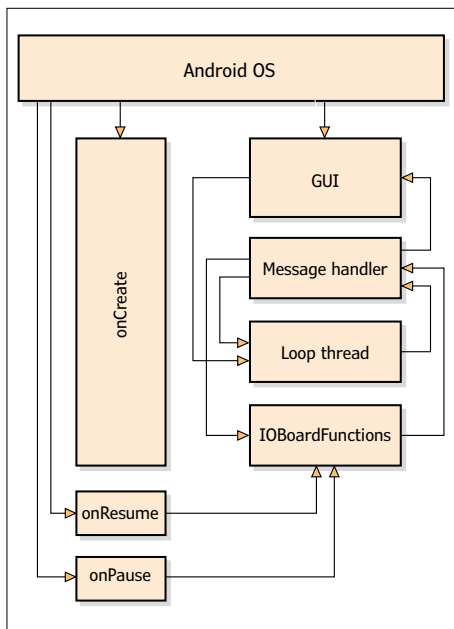


Figure 3. The structure of the example app.

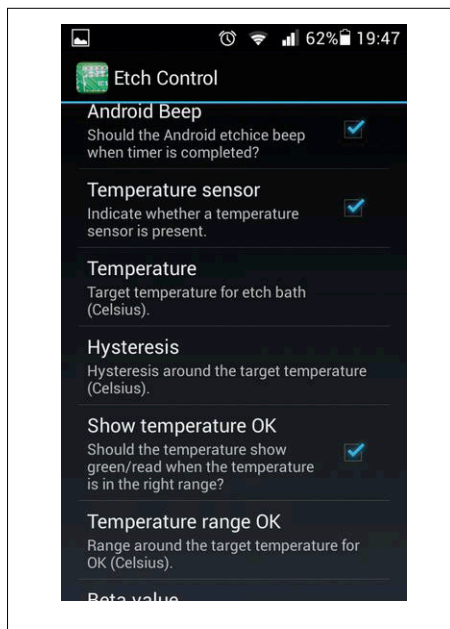


Figure 4. Various settings in the etching app.

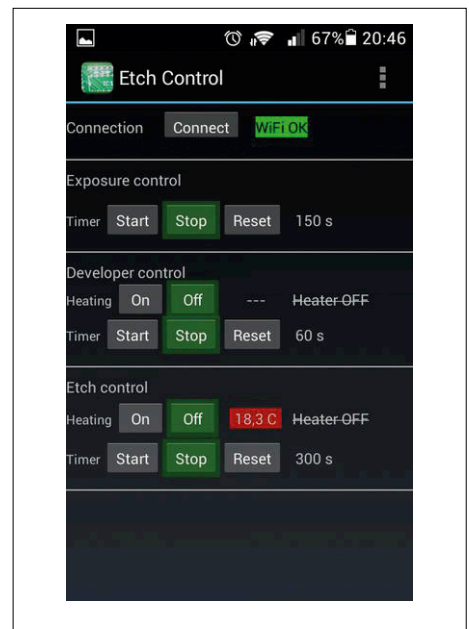


Figure 5. The control surface of the app in portrait mode.

pletely independent. It is therefore not possible (nor is it advisable) to have the code write to the stream and then wait for a reply from the Android I/O board in some way. Instead, the reply of the controller is analyzed in something known as a message handler. This message handler is (indirectly) called by Android at the instant when data has been received. This is represented in **Figure 3** by the arrows. The data is then analyzed by inspecting the first few characters (for example, 'G B0' is followed by the value read from B0). From inside the message handler you can change graphical elements or call functions that do this.

The app also has to periodically read the values from the Android I/O board. This can't be done from the main thread. With Android programs it's the case that while the code in the main thread is running, the graphical user interface (GUI) can't be updated. More importantly, when the main thread has run for more than a few seconds, Android assumes that it has crashed and will ask the user if it's ok to close it. For these reasons, you have to create a separate thread that runs outside of the main thread, which periodically deals with the readings in an endless loop. It's not possible to update the graphical elements of the GUI from within the thread (to update some text, for example). This also has to be done via a message handler.

The communications from the main thread to the thread (when a button has been pressed on the GUI, for example) can also take place via a message handler. However, in this app we've used a variable as a flag. Since the thread is in an endless loop, it can read the variable and perform specific actions, depending on its value.

The thread must not be running continuously, since it would then take up all the CPU time and the Android device would become unusable. For this reason you have to put a 'Sleep' in the loop of the thread, which causes the thread to 'sleep', giving the CPU time to deal with all the other tasks. In this app a sleep time of 25 ms has been used, as a compromise between CPU load and respon-

siveness when data from the Android I/O board has been received. When you're writing your own app, this period may be longer or shorter, depending on your requirements.

A command is sent once per second from inside the loop that requests the temperatures (from the NTCs), using 'r b3' and 'r b4'. The replies, 'R B4 x' and 'R B5 x' respectively (where x is the decimal value), are sent to the message handler with the tag 'RECEIVE_MESSAGE_CONDATA'. In the message handler the first four characters are compared with 'R B4' and 'R B5'. If they match, the rest of the string is converted into a temperature and stored in a variable (refer to the inset to see how the temperature is derived from the measured value of the NTC). Another variable is then set to indicate

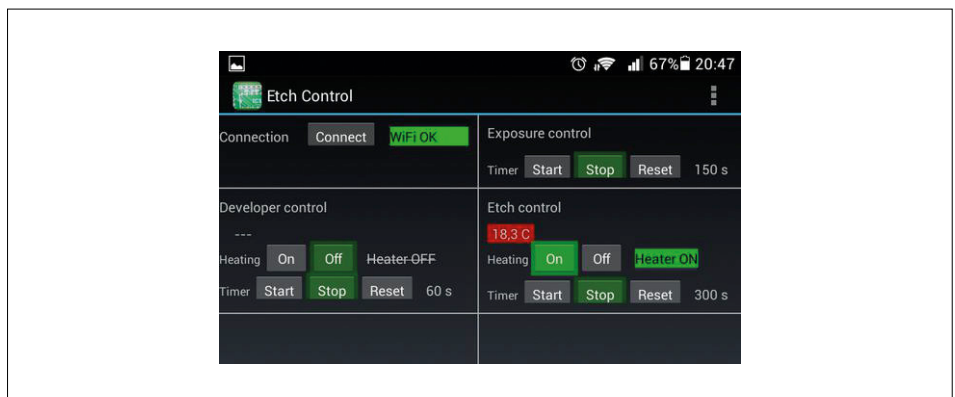


Figure 6. The same menu, but now in landscape mode.

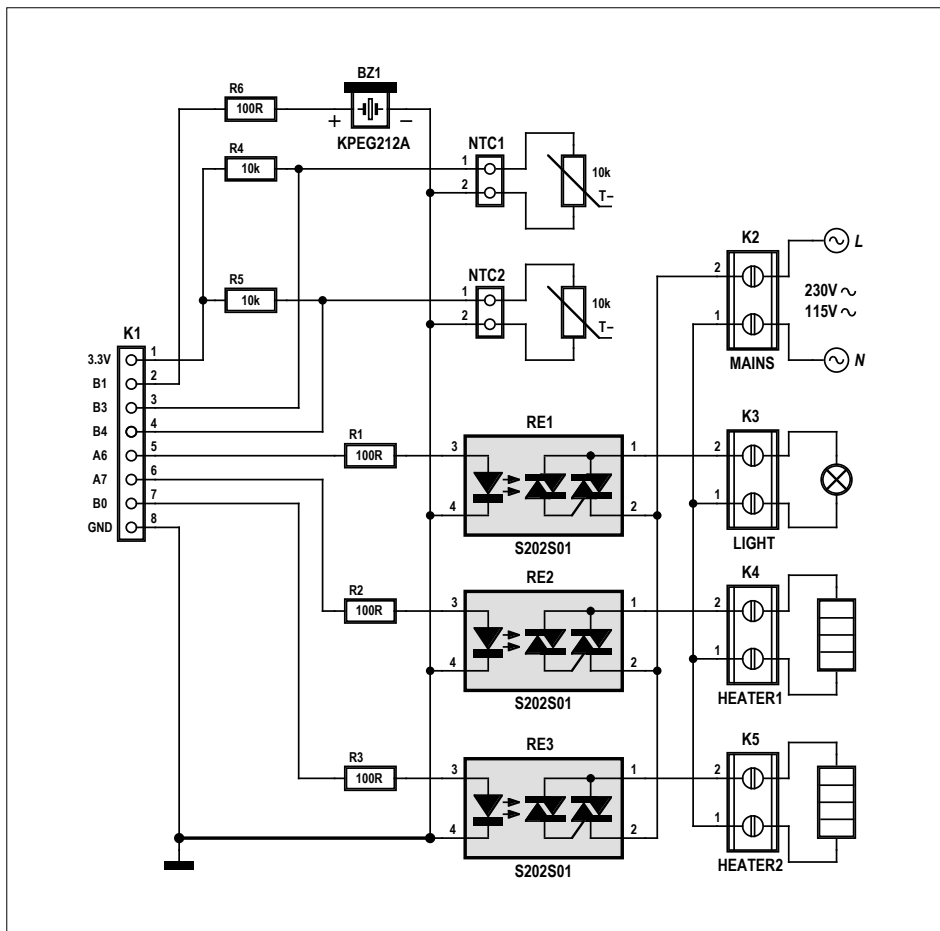


Figure 7. The circuit for controlling the light box and for heating the developer and/or etching tanks.

that a new value has been received. This is detected in the loop and a decision is made whether the heating element should

be turned on or off. The result is passed via the message handler to functions in the main thread, which update the GUI

and control the Android I/O board (for example: 'W B0 0' to turn the heating element off).

The app also uses the watchdog function on the Android I/O board. When the board hasn't received any data for more than a minute, it will automatically turn off the outputs to the heating elements. The temperature measurement sends data once per second. So when the app has stopped or the connection has been lost, it will automatically avoid overheating. If you want to personalize the app, you can modify the addresses of the Bluetooth, WiFi or USB modules, the temperatures, periods for the timers and graphical elements in the settings menu. The settings menu is defined in the res/xml/user_settings.xml file and has its own class (UserSettingActivity) in the file src/com.Example.EtchControl/UserSettingActivity.java. This is called from the main thread using 'startActivity(new Intent(-MainActivity.this, UserSettingActivity.class))'. The values for the settings are automatically stored in a local database and can be retrieved using the preference manager.

The settings menu is called from the menu that is defined in src/menu/menu.xml. In java this xml file is 'loaded' using getMenuInflater().inflate(R.menu.main, menu);.

Android has the facility to have different definitions for the layout for the user-in-

Component List

Resistors

R1,R2,R3,R6 = 100Ω
R4,R5 = 10kΩ

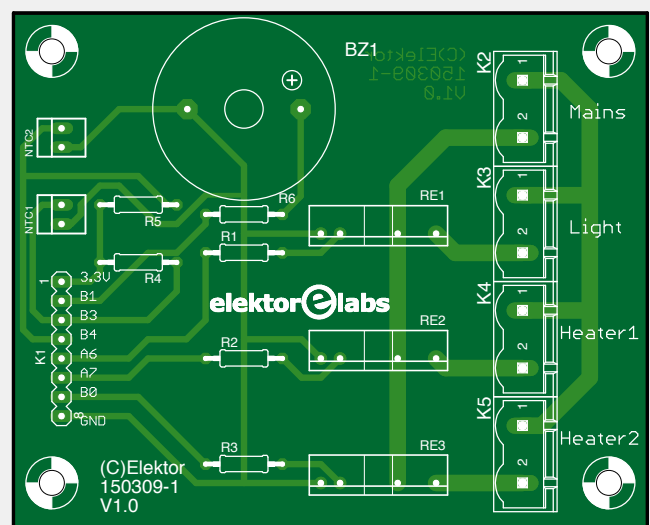
Semiconductors

RE1,RE2,RE3 = S202S01 solid-state relay

Miscellaneous

BZ1 = active piezo buzzer
K1 = 8-pin pinheader, 0.1" pitch
NTC1,NTC2 = 2-pins pinheader, 0.1" pitch
4 pcs 2-way PCB terminal block, 0.2" pitch, 230VAC rated, LIGHT, HEATER1 and HEATER2
2 pcs NTC 10kΩ

Figure 8. The board layout for the circuit in Figure 7. Take great care, since part of the board is connected directly to the live AC line!



interface, depending on the orientation of the screen (portrait or landscape mode). The standard layout (**Figure 5**) is in layout/layout.xml. When you create an alternative layout in layout-land/layout.xml, Android will automatically use this when the screen is in landscape mode (**Figure 6**).

The circuit

In **Figure 7** you can see the circuit diagram of the hardware that measures the temperatures and controls the light box and the heating elements. You can connect two NTCs, one for the temperature measurement of the developing tank, the other for that of the etching tank. Each NTC (NTC1/NTC2) is part of a resistor network, using resistors R4/R5. The junctions of these networks are connected to pins B3 and B4, respectively.

The 230 V heating elements are controlled by solid-state relays, since mechanical relays that operate from 3.3 V are very hard to come by. The S202S01 used here can switch up to 8 A. When you switch a current of more than a few amps you

will need to fit a heatsink to the relay. The relays are connected to pins A6, A7 and B0. The author only had a heated etching tank and a light box, so only two of the relays had to be mounted. This can be specified in the app. The author did measure the temperature of the developing tank with the other NTC, since it can be too warm immediately after mixing, or too cold if it's been out in the shed. The buzzer is connected to pin B1. This is a piezo-electric type with built in electronics.

The author has designed a single-sided board for this small circuit, which is shown in **Figure 8**. The layout and the Eagle files

can be downloaded from [4]. If you're going to switch more than several hundred watts, you must reinforce the tracks by tinning them or adding wires in parallel. The circuit could also be built on an experimenter's board, but then you have to make absolutely sure that it is electrically safe. In any case, the circuit should be mounted in a fully isolated plastic enclosure.

In the third installment we'll take a look at the data logging function of the Android I/O board and explain how the firmware can be modified and how to upload it using the bootloader. ◀

(150309)

Web Links

- [1] <http://ww1.microchip.com/downloads/en/DeviceDoc/50002230A.pdf>
- [2] <http://tssh2.sourceforge.jp/index.html.en>
- [3] www.ftdichip.com/Drivers/D2XX.htm
- [4] www.elektormagazine.com/150309
- [5] <http://developer.android.com/tools/help/adt.html>
- [6] <http://developer.android.com/sdk/index.html>

Advertisement

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

- 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



Only \$28.95!

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



www.dlpdesign.com

The Easiest Way to Design Custom
Front Panels & Enclosures

Free
Front Panel
Designer



You design it
to your specifications using
our FREE CAD software,
Front Panel Designer

We machine it
and ship to you a
professionally finished product,
no minimum quantity required

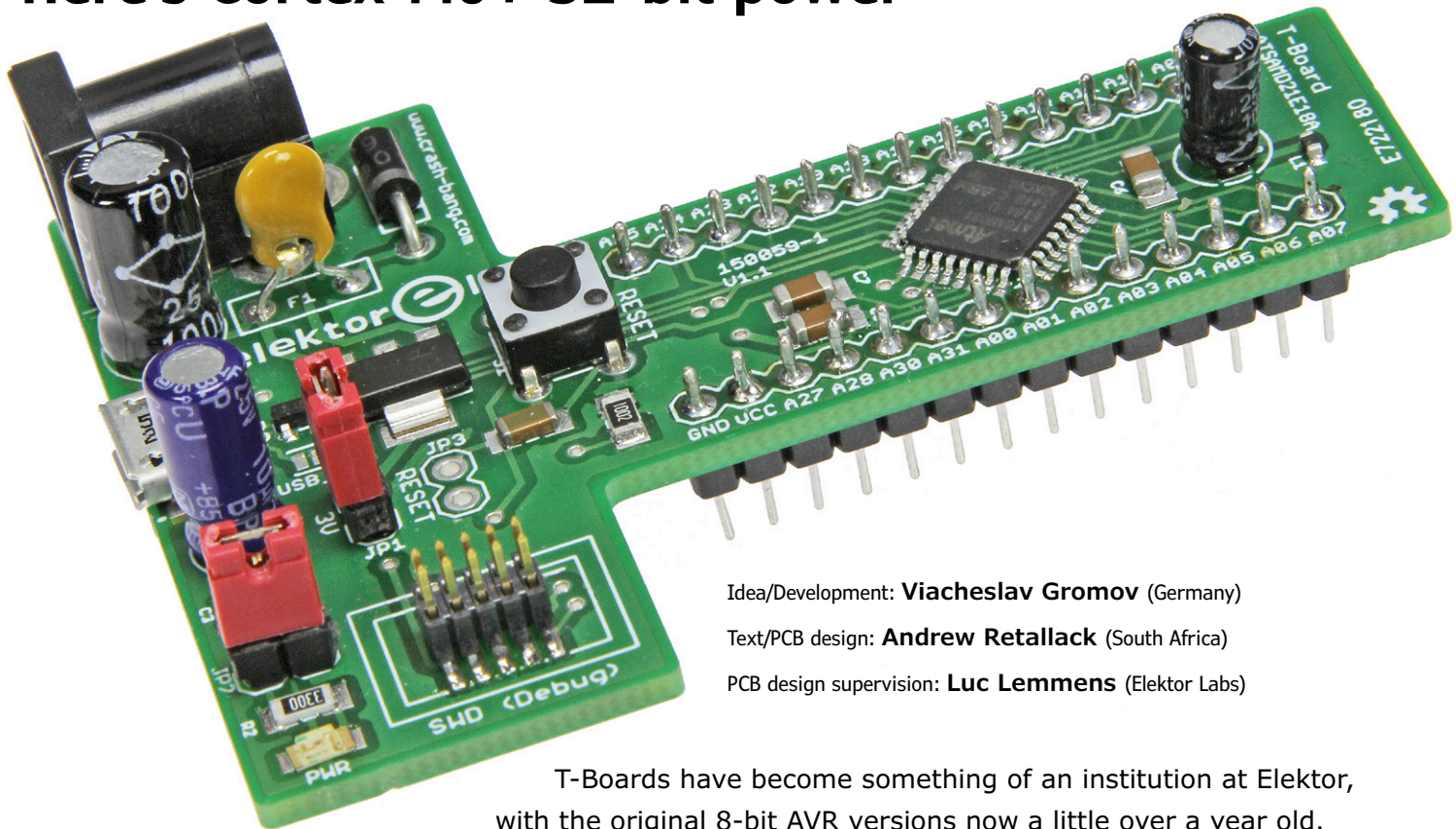
- Cost effective prototypes and production runs with no setup charges
- Powder-coated and anodized finishes in various colors
- Select from aluminum, acrylic or provide your own material
- Standard lead time in 5 days or express manufacturing in 3 or 1 days

**FRONT PANEL
EXPRESS**

FrontPanelExpress.com

ARM'ed T-Board

Here's Cortex-M0+ 32-bit power



Idea/Development: **Viacheslav Gromov** (Germany)

Text/PCB design: **Andrew Retallack** (South Africa)

PCB design supervision: **Luc Lemmens** (Elektor Labs)

T-Boards have become something of an institution at Elektor, with the original 8-bit AVR versions now a little over a year old.

Since then we've also seen wireless and audio T-Boards. Now we introduce the youngest (and most powerful) member of the family — meet the 32-bit ARM T-Board.

While 8-bit microcontrollers remain the bread and butter of the enthusiast and maker communities, there is a growing interest in 16- and 32-bit microcontrollers. Without doubt, the most popular emerging architecture is the ARM Cortex-M — a core that nearly all the major manufacturers include in their microcontroller line-ups. Unquestionably, the popularity is in part driven by declining prices (you can now buy a small Cortex-M0+ for less than \$1), combined with a requirement for more processing power, features and I/O pins as interest in IoT continues to grow. Also, with Elektor Magazine running its "From 8 to 32 Bits" ARM programming course, the educational aspect is covered too.

While there are a range of ARM Cortex development boards around, we believe that the T-Board is a perfect platform for an ARM Cortex-M0 microcontroller. ARM chips come in small-pitch packages, and

need a range of supporting components, making them not that "user-friendly" straight off the shelf — they need to be mounted on some kind of development board to be at all useful to the enthusiast, or for prototyping purposes. Sadly the existing development boards are not designed for use on breadboards, making them sometimes clumsy and hard to use in a prototyping environment. Enter the T-Board: specifically designed to be plugged directly into the breadboard with all the pins brought out and unobscured. If you've ever used another T-Board you'll know just how quick and easy it is to get up and running with a prototype!

Arming the T-Board

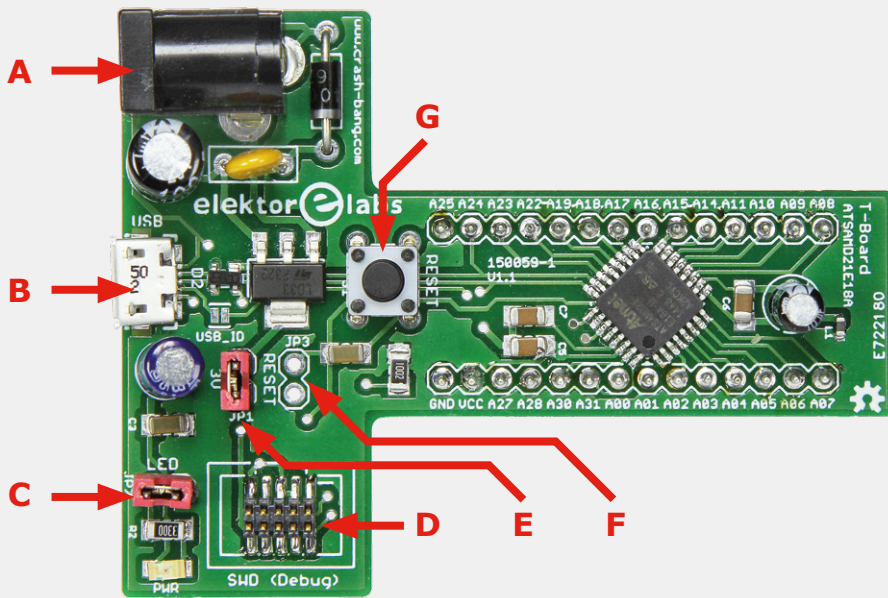
The ARM-equipped T-Board houses the Atmel ATSAMD21E18A microcontroller, an ARM Cortex-M0+. If you aren't familiar with the ATSAMD21E18A, take a quick look over these specifications:

- 8/16/32-bit Timer/Counter
- 3 24-bit Timer/Counters
- 32-bit RTC with clock/calendar functions
- USB 2.0 interface (host and device functionality)
- 4 Serial interfaces (USART, I²C, SPI, LIN)
- 12-bit DAC with 20 channels (differential and single-ended input; oversampling support in hardware)
- 10-bit DAC
- Inter-IC Sound (I²S) Interface
- 2 Analog Comparators
- 16 External Interrupts
- 26 GPIO pins
- Peripheral Touch Controller: 10 x 6 lines
- Maximum frequency: 48 MHz
- 1.62 V to 3.63 V supply

That's quite a handful of features compared to *ye olde 8-bit micro*. We felt that

Physical Layout of Board

- A. Power Connector:** 2.1mm center-positive DC Jack (max. 9 V)
- B. Micro USB Connector**
- C. LED Jumper:** Connect/Disconnect the power LED
- D. SWD Header:** 10-pin programming header (Serial Wire Debug)
- E. Current Measurement:** Remove jumper to enable in-line current measurement
- F. Reset Header:** Enables remote reset of board
- G. Reset switch**



ID, is by default not connected to the MCU (in order to free up available pins). A solder jumper however allows you to connect it to pin PA03. The ID pin is used for USB On the Go (OTG); refer to Atmel's application note for more details [1].

An additional feature is a 2-pin header on the Reset line: this enables remote resetting of the microcontroller. Programming and debugging are through a 10-pin 0.05-inch pitch SWD header. SWD (Serial Wire Debug) is a subset of the JTAG interface, and only uses 6 wires. The 10-pin header is there to maintain compatibility with Atmel programmer/debuggers such as the Atmel ICE.

The rest of the board essentially contains the headers that break the microcontroller's I/O pins out, and a collection of supporting components: a healthy dose of decoupling capacitors as specified in the datasheet, as well as a ferrite bead (L1) to prevent VDD noise interfering with the analog supply VDDANA.

You'll note that there is no quartz crystal on the board. This was to give the user flexibility to choose an external crystal that suited their design best: either a crystal oscillator in the range 0.4 MHz to 32 MHz on PA14/PA15, or a 32.768-kHz watch crystal on PA00/PA01 for RTC functionality.

Reducing power consumption

You may recall from the *T-Boards Lowest Power Exercising* article back in December 2014 that the author wrote an article on ways to measure and then reduce

This T-Board is a perfect platform for an ARM Cortex-M0+ microcontroller

Component List

Resistors

SMD 1206, 1%, 0.125W
 R1 = 10kΩ
 R2 = 330Ω

Capacitors

C1 = 100μF 25V 20%, radial
 C2, C8 = 10μF 25V 20%, radial
 C3, C4, C5, C6, C7 = 100nF 50V, X7R, 1206

Inductor

L1 = BLM18PG471SN1D ferrite bead, 0.2Ω/1A, 470Ω @100MHz

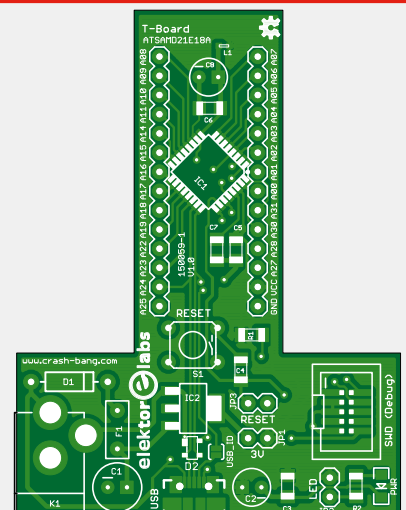
Semiconductors

D1 = 1N4007 (1000V, 1A)
 LED1 = LED, red, SMD 1206
 IC1 = ATSAM21E18A-AU
 IC2 = LD1117S33CTR (3.3V LDO voltage regulator)
 IC3 = PRTR5V0U2X (ESD protection diode)

Miscellaneous

S1 = switch, tactile, 24V, 50mA, 6x6 mm
 JP1, JP2, JP3 = jumper, 1x2, vertical, 0.1" pitch
 F1 = 500mA PTC resettable fuse
 K1 = DC barrel jack 2.1mm pin
 K2, K3 = 14-way pinheader, SIL, 0.1" pitch
 K4 = 10-way (2x5) pinheader, 0.05" pitch
 K5 = micro USB receptacle, 2.0 type B, SMD jumper socket 0.1" pitch
 PCB, Elektor Store # 150059-1
 Alternatively: ready-assembled board, Elektor Store # 150059-91

Figure 2. Component layout of the ARM'ed T-Board. Don't worry, the board is available ready manufactured from the Elektor Store.



the power consumption of the T-Board-28 (housing an ATmega328 microcontroller). The ability to achieve this kind of flexibility and control was one of the key motivations behind the design of the T-Board — and this has not been lost in the design of the T-Board ARM.

Jumper JP1 enables you to connect current-sensing circuitry to measure the consumption of the board, and therefore to assess the impact of any code changes and optimizations introduced to reduce the current draw. The original AVR T-Boards allowed you to do this; however we have improved on this in the T-Board ARM by including JP2. That jumper allows you to disconnect the power LED, so that the draw from the LED doesn't skew your measurements.

The physical board design

If you've already used a T-Board you'll see the same heritage running through this board's blood-lines. You will recall that the horizontal cross-bar of the "T" is designed to keep the bulk of the components out of the way of your breadboarding, while the vertical stem breaks the microcontroller's pins out in a way that makes it straightforward to plug into the breadboard and access them.

The T-board is of course open source in the spirit of sharing, and the design files are downloadable for you if you have the equipment and wherewithal to make your own board [3] using the parts list and the board layout in **Figure 2**. If you aren't feeling quite brave enough to tackle the 0.8-mm pitch of the ARM microcontroller TQFP package, or the invisible pins of the USB connector, then the board is available through the Elektor store, fully assembled and tested, and ready to be plugged into your breadboard.

ARMed-'n-ready

I'm sure you've heard enough about the design and want to see the board in action! Let's keep this project straightforward to illustrate the use of the board. To explore more complex projects, turn back to the series *From 8 to 32 bits: ARM Microcontrollers for Beginners* that made its debut in the January & February 2015 edition of Elektor Magazine.

Working with ARM microcontrollers is definitely more complex than with AVR microcontrollers. If you're new to them they require perseverance and some time

spent reading up on their systems and architecture. Our purpose here is to introduce you to the T-Board, not to dive into the intricacies of ARM microcontrollers — we don't have nearly enough space in this article, and the "From 8 to 32 bits" series covers it comprehensively. Moreover, Viacheslav Gromov who kicked off this project will be covering software specifically on this ARM T-Board in a future edition. With these *caveats*, let's get cracking.

Step 1: Layout the Breadboard

The first step is to place the T-Board onto a breadboard. Position it so that the section containing all the power components projects over the edge of the breadboard and the header pins are located on either side of the breadboard's central channel. Then:

1. Connect a jumper between the GND pin and the negative power rail of the board.
2. Connect the anode of an LED to pin A07, placing the cathode in an empty row on the breadboard.
3. Place a resistor so that one leg is in

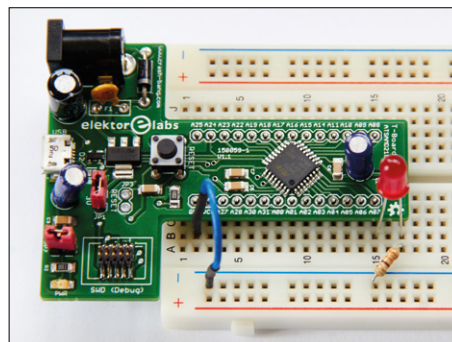


Figure 3. This ARMed-T-Board-on-a-breadboard is all ready to go.

the same breadboard row as the cathode, and the other leg inserted into the negative power rail.

Figure 3 shows the board plugged onto a breadboard and all connected up — I'm sure you'll agree that it's impressive to see only one jumper on the entire board!

Step 2: Create the Project

Next up is the creation of the project. We'll work in Atmel Studio 6.2 for this

Alternative IDEs

When you start working with ARM microcontrollers, you're playing in the "big league" when it comes to Integrated Development Environments. There are a range of developers who provide what seem to be fairly solid tools — but it's out of the scope of this article unfortunately to draw full comparisons between what are complex pieces of software. To give you a taste of what's out there, here are three of the more popular ones.

Atmel Studio 6.2

This in my mind is the first choice for those not professionally developing on ARM microcontrollers. It is free, is not code-limited, and is written and supported by Atmel themselves, hence is guaranteed to support not only the microcontrollers but also the Atmel programmers and debuggers. www.atmel.com/atmelstudio

IAR Embedded Workbench for ARM

IAR have a wide range of professional-level IDE's — and of course at professional prices! They support Atmel's AVR and ARM microcontrollers, as well as a range of ARM and non-ARM MCUs from other manufacturers. If you work with a range of microcontrollers from different manufacturers, then this may be a good bet if you can afford it. There are code-limited versions, but so limited that they are more useful for assessing the tool than for any practical projects. www.iar.com/iar-embedded-workbench/arm

Keil

Keil MDK-ARM is an IDE that is owned by ARM itself, and is a respected tool for development of ARM projects — primarily aimed at professionals. However, Keil do offer a "lite" edition of their MDK-ARM tool that is code-limited to 32 KB — probably enough for most enthusiasts and hobbyists. If you have the time to spend, and want to explore ARM micros from other manufacturers, then it's well worth looking at. www.keil.com

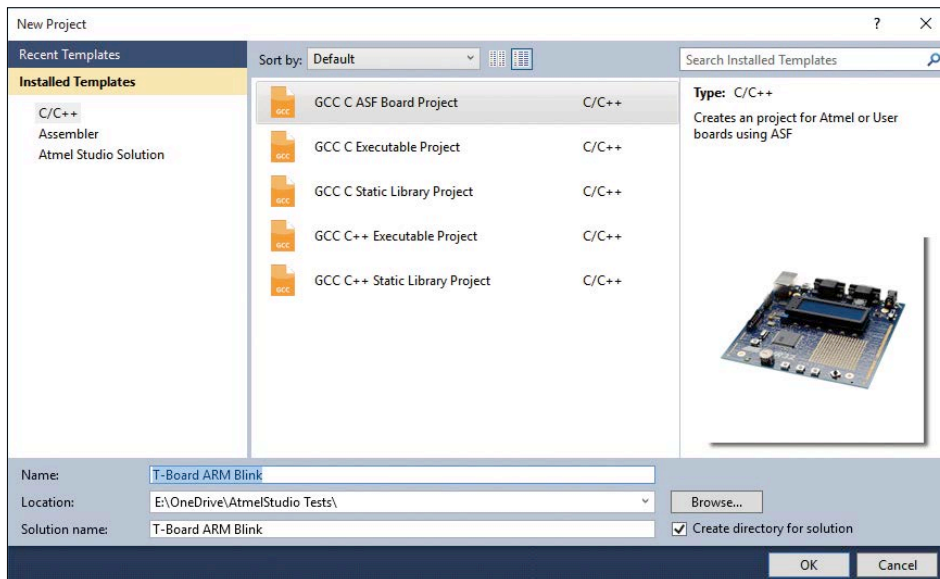


Figure 4. Initial dealings with the GCC ASF Board Project.

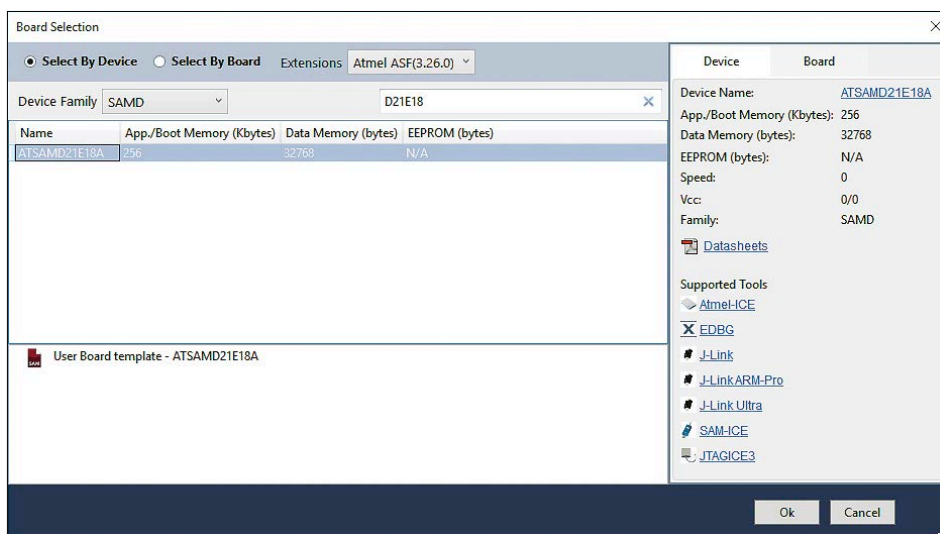


Figure 5. Selecting a non-AVR micro.

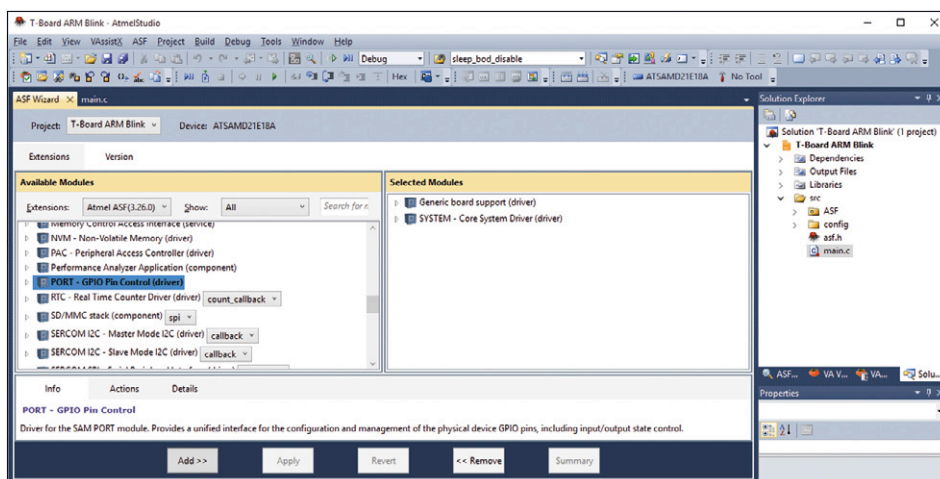


Figure 6. Here you add the PORT driver.

(see **inset**: Alternative IDEs), and using the Atmel Software Framework (ASF), which we found the quickest way to get up and running given the complexity of an ARM micro. In the “From 8 to 32 bits” series ASF is also used, although with an Atmel SAMD20 XPlained Pro board. We have to do things slightly differently to set the project up here, but from there on it’s fairly straightforward.

Follow these steps:

1. Open Atmel Studio, and create a new project — choose the *GCC ASF Board Project*, name it and choose where to save it (**Figure 4**).
2. Select the device: The T-Board ARM uses an *ATSAMD21E18A*, so choose that. As there aren’t any Atmel development boards for this microcontroller, you must select *User Board Template* and click OK (**Figure 5**).
3. An empty project has been created, with *main.c* under the *src* folder in the Solution Explorer.
4. Next we need to add the ASF PORT driver to the project. This provides a series of functions to allow us to access the pins more easily. Choose *ASF Wizard* from the ASF menu.
5. On the popup dialog (**Figure 6**) you’ll see available modules on the left, and selected modules on the right. Scroll down the left and highlight *PORT – GPIO Pin Control (driver)*, then click Add and Apply. You’ve now added the PORT driver to your project.
6. Now add the code from **Listing 1** into the *main.c* module.

Step 3: Flash the T-Board

Your project is now ready to be compiled and uploaded to the T-Board:

1. Compile the project (F7), ensuring that there are no errors of course.
2. Connect power to the board through the DC jack (assuming your programmer doesn’t provide power to the project). The Atmel ICE that I use is one of these that doesn’t power the target. If you use a JTAGICE3 programmer then be sure to have relevant information on using it.
3. Attach the programmer to the SWD header on the board, and then connect to the USB port on the PC.
4. Select the programmer from the *Tools* menu, under *Device Programming*.

Web Links

- [1] Atmel USB 2 GO: www.atmel.com/Images/Atmel_11201_USB-OTG-Like-Connector-Implementation_SAM9G-SAM9X-SAMA5D3_Application-Note.pdf
- [2] T-Boards low power article: www.elektormagazine.com/140413
- [3] Project resources page: www.elektormagazine.com/150059
- [4] Atmel Studio: www.atmel.com/tools/atmelstudio.aspx
- [5] www.der-hammer.info/terminal/

5. Finally upload the program: choose *Start without Debugging* from the *Debug* menu.

6. You should now see a blinking LED!

The next steps

Now that you've seen the T-Board in action, you're probably chomping at the bit to start additional projects. If you're new to ARM Cortex microcontrollers, I'd

challenge you to take one of the previous projects in the "From 8 to 32 bits" series and use the T-board in place of the Xplained Pro board. You'll cut down the number of jumpers, making the projects easier to design and troubleshoot.

Meanwhile, co-designer Viacheslav here presents another kick-off program specially written on the occasion of this "hardware" dominated article: see **List-**

ing A and the **inset:** My/Your/The/A First Project on the next two pages.

We hope that you enjoy using the ARMed T-Board both as a learning tool and a way to get your next project prototyped and running more quickly! ◀

(150059)

Mind U! ...continued overleaf

Listing 1. Blink-a-LED (in ARM T-Board Style)

```
#include <asf.h>

#define LED_PIN PIN_PA07 //LED is connected to PA07

//Function Prototypes
void configure_port_pins(void);

int main (void)
{
    system_init(); //ASF Routine to initialise the system, clocks, etc.

    configure_port_pins(); //Configure GPIO pins

    SysTick_Config(system_gclk_gen_get_hz(GCLK_GENERATOR_0)); //Enable SysTick interrupt:
                                                                //reads frequency to sets interrupt at 1 Sec

    while (1)
    {
        //Nothing needed here as we are using interrupts to toggle the LED
    }
}

//Function to configure the port pins
void configure_port_pins(void)
{
    struct port_config config_port_pin; //Structure used to store parameters
    port_get_config_defaults(&config_port_pin); //Read the current configuration of the pins into the Struct
    config_port_pin.direction = PORT_PIN_DIR_OUTPUT; //Set Struct to indicate pin is an Output
    port_pin_set_config(LED_PIN, &config_port_pin); //Use the struct to set the direction for the LED's pin
}

//Interrupt Handled for SysTick Interrupts
void SysTick_Handler(void)
{
    port_pin_toggle_output_level(LED_PIN); //Toggle the LED's Pin
}
```

My/Your/The/A First Project

By Viacheslav Gromov

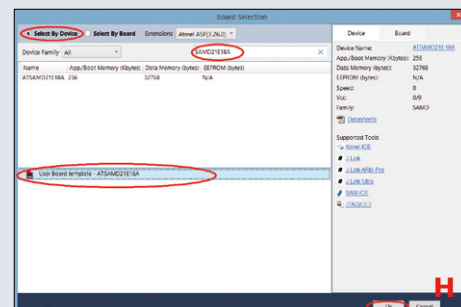
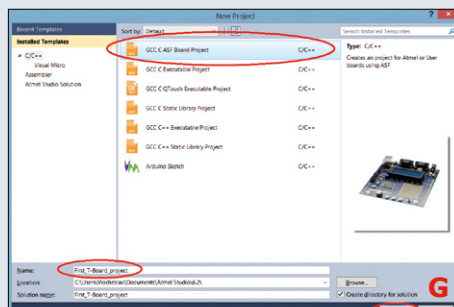
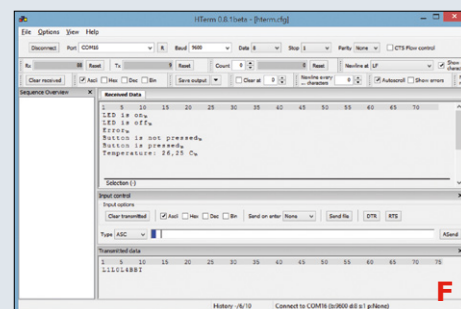
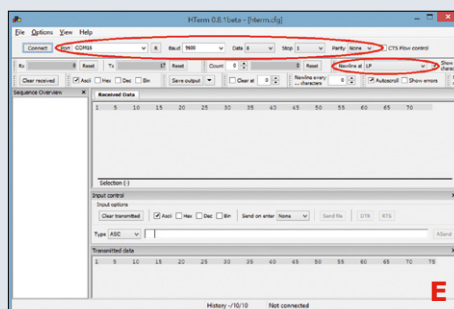
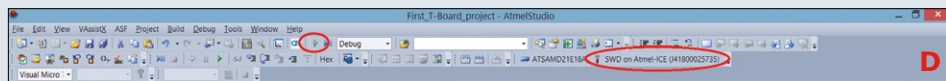
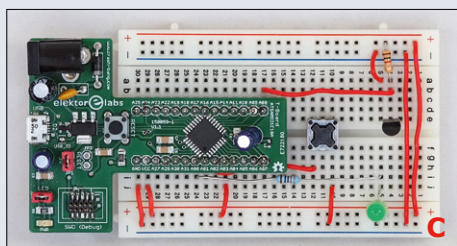
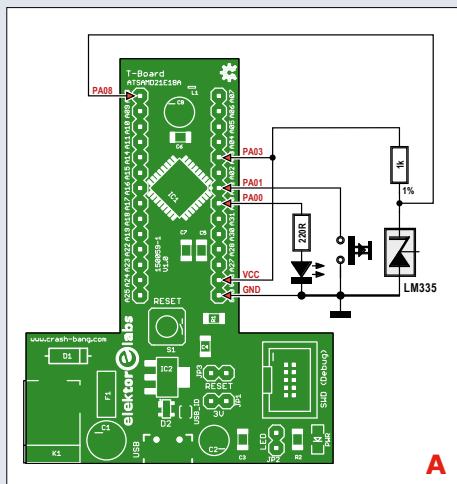
Let's make a simple project! Let's switch one LED on and off, sample one pushbutton, and measure the temperature with an LM335 and ADC. We want to control everything from the PC through USB with a no-frills terminal emulator program.

1. The Hardware

Figure A shows how the external parts connections to the ARM'ed T-Board, **Figure C**, the practical realization. With all parts connected up, you can use a USB-A connector to USB Micro-B connector cable to connect the T-Board to your PC. This type of cable is often used for charging smartphones or tablets, so you should have one ready to hand. You also need to connect a debugger like Atmel ICE to the T-Board on the SWD (Debug) connector and of course also to your PC (**Figure B**).

2. The Software

First, download the Atmel Studio project named "First_T-Board_project" from [3].



Next, open it in Atmel Studio 6, which you can download under [4] after a minor registration or as a guest, if you haven't installed it yet. If you are not familiar with the Atmel SAM D family and/or Atmel Studio look in our *ARM Microcontrollers for Beginners* course.

The program starts with the declarations of the variables and arrays, function prototypes and configuration-functions for the peripherals. After that, in the main function, the system, the interrupts and the delay-functions get initialized and all the configuration-functions get called.

Listing A shows the endless loop with the main part of code. The whole code is in a switch statement which receives one ASCII-character with `udi_cdc_getc()` on the USB and chooses the correct branching to follow. In **Table A** you can see all characters (combinations) you

can send later to the board with the terminal program. For each of the three letters there is a case. The number of the case isn't the letter, it's the ASCII-number for the letters: 76 stands for "L", 66 for "B" and 84 for "T". Please note that these are all capital letters.

In the first case, 76, there is an `else-if` statement which turns the LED on, when variable number is 1, and off, when number equals 0. The variable number gets the (second) character from the USB with `number = udi_cdc_getc()`; before this. Also, "LED is on" and "LED is off" will be sent with `udi_cdc_write_buf(&buffer, x)` over the USB. For arguments it needs only a pointer to a string with the data (buffer) it has to send, and the number of characters due sending (x). But here, we don't use a pointer to a buffer — to keep things simple, we embed a text like "LED is on"

Table A. Mini Command Set

L	1	Sending 'L1' (for: LED) causes the LED to be switched on and you receive a message "LED is on". Sending 'L0' switches the LED off and you get "LED is off". Sending 'L' with a number other than 0 or 1 returns: "Error".
	0	
B		Sending "B" (for: Button) causes the T-Board to respond with "Button is pressed" or "Button is not pressed".
T		Sending "T" (for: Temperature), causes the T-Board to emit a string with the currently measured temperature x, like: "Temperature: x C".

direct into the command like this: `udi_cdc_write_buf("LED is on\n", 10);`. The `\n` stands for new line. If this variable is found to contain another value than one or zero, "Error" will be sent to the computer.

In the next case, 66, an `if` statement processes the state of the button pin, PA01, with: `port_pin_get_input_level(PIN_PA01)` and sends either a "Button is pressed" or "Button is not pressed" message over the USB.

The last case, 84, measures the voltage on ADC channel 16 (PA08) supplied by the LM335, and calculates the temperature from this voltage. A string like "Temperature: x C" gets sent over USB in a more circumstantial way due to the specific requirements of the float-temperature-value described in the ARM Course.

3. The First Test

Let's try it out! Please select your debugger above on the right and start compiling and transfer the program with a click on the green "Start Without Debugging"-button (**Figure D**). After the MCU is programmed, the T-Board will register on your computer as a "Communication Device Class ASF example" – yep, a virtual serial interface. Maybe your computer needs some time to install the right drivers. After it's done, you can open the device manager on the PC to see which COM port number's been assigned to your T-Board under "Ports (COM & LPT)". Now open a terminal program like HTerm [5], and set up the comms like in **Figure E**:

- COM port number (see Device Manager)
- 9600 baud
- 8 data bits
- 1 stop bit
- no parity
- new line on LF

With all settings adjusted and DTR On, connect with the board and test it with our command-capital letters like in **Figure F**. Congratulations, your first project with the ARM'ed T-Board is running now! That was all copycat though – now on with building a new project.

4. Create a new project with the ARM'ed T-Board & Atmel Studio

The project generation is nearly the same

as described in the first part of the ARM course. You also need to select a "GCC C ASF Board Project" under File/New/Project... (**Figure G**), but in the next step, after you pressed "OK", it's different – you now need to select our MCU (ATSAM-

D21E18A) by Device like in **Figure H**. After you pressed "OK" once more the c project is generated. Now you can import the ASF libraries you need in the ASF Wizard and write your code!

Listing A. The endless loop of our get-u-going program looks like this.

```
while(1){
switch (udi_cdc_getc())
{
    case 76: //if received "L"
        number = udi_cdc_getc(); //get the second character (a number)
        if (number == 49) //if received a one
        {
            port_pin_set_output_level(PIN_PA00, 1); //put the LED on
            //send "LED is on" with a "new line" on USB
            udi_cdc_write_buf("LED is on\n", 10);
        }
    else if(number == 48) //if received a zero
    {
        port_pin_set_output_level(PIN_PA00, 0); //put the LED off
        //send "LED is off" with a "new line" on USB
        udi_cdc_write_buf("LED is off\n", 11);
    }
    else //if received a wrong number (not 1 or 0)
    {
        udi_cdc_write_buf("Error\n", 6); //send "Error" and a "new line" on USB
    }
    break;
    case 66: //if received "B"
        if(!port_pin_get_input_level(PIN_PA01)) //get the level on PA01 (button)
        {
            //if pressed, send "Button is pressed" and a "new line" on USB
            udi_cdc_write_buf("Button is pressed\n", 18);
        }
    else
    {
        //if not pressed, send "Button is not pressed" and a "new line" on USB
        udi_cdc_write_buf("Button is not pressed\n", 22);
    }
    break;
    case 84: //if received "T"
        adc_start_conversion(&adc_instance); //start ADC-conversion
        //read and save the conversion result
        while(adc_read(&adc_instance, &data) == STATUS_BUSY){}
        //calculate the temperature
        temperature = (25 + (data * 0.000805 - 2.945) / 0.01) * 100;
        //reformat the result in a buffer
        sprintf(temperature_string, "%i", temperature);
        //send "Temperature:" with USB
        udi_cdc_write_buf("Temperature: ", 13);
        //send the temperature on USB
        for(i = 0; i < 4; i++){
            if((temperature >= 1000) && (i == 2)) udi_cdc_putc(44); //make a comma
            if((temperature < 1000) && (i == 1)) udi_cdc_putc(44); //make a comma
            udi_cdc_putc(temperature_string[i]); //send a digit on USB
            //send "C" and "new line" on USB at the end of transmission
            if(i == 3) udi_cdc_write_buf(" C\n", 3);
        }
    break;
}
}
```


Compact 60-watt Amplifier

Cheap & cheerful design packs a punch



By **Dr. Robert M. Carter** (United Kingdom)

It may not be the highest of Hi-Fi but at only 38x56 mm this low-cost discrete design is smaller and cheaper than IC modules of similar power, yet still delivers excellent sound quality, making it ideal for in-car audio, domestic subwoofers, smartphone/tablet speakers or multi-channel AV...

Building your own amplifier is not as popular as it used to be, possibly because sophisticated designs with exotic audio transistors can work out as expensive as ready-made stacking separates from China. Of course, changes in fashion influ-

ence the market too; do-it-yourself amplifier designs were popular in the disco era, when there were cost savings to be made. Perhaps then, low cost is the key to making home construction attractive. Enter the 60-watt compact amplifier, orig-

inally designed 15 years ago to be a small and cheap discrete-transistor module that could pack a healthy punch for its size. The circuit was designed around the best cost/power ratio transistors available at the time, the BDW93C/94C Darlingtons,

Specifications

- Input sensitivity: 0.2 V_{rms} (alternatively 1V_{rms}; see text)
- Input impedance: 7 k Ω (or 14 k Ω with lower gain option; see text)
- Sinewave power: 4 Ω 60 W, THD+N = 1%
- 8 Ω 36 W, THD+N = 1%
- Bandwidth: 20Hz – 80kHz (-3dB)
- Signal to noise ratio: >90 dBA
- THD + noise: 0.045% at 1 kHz (10 W/4 Ω)
- Damping factor: >250 (1 kHz)
- (>1300 at lower gain – see text)
- Quiescent current: \approx 33 mA (complete amplifier)

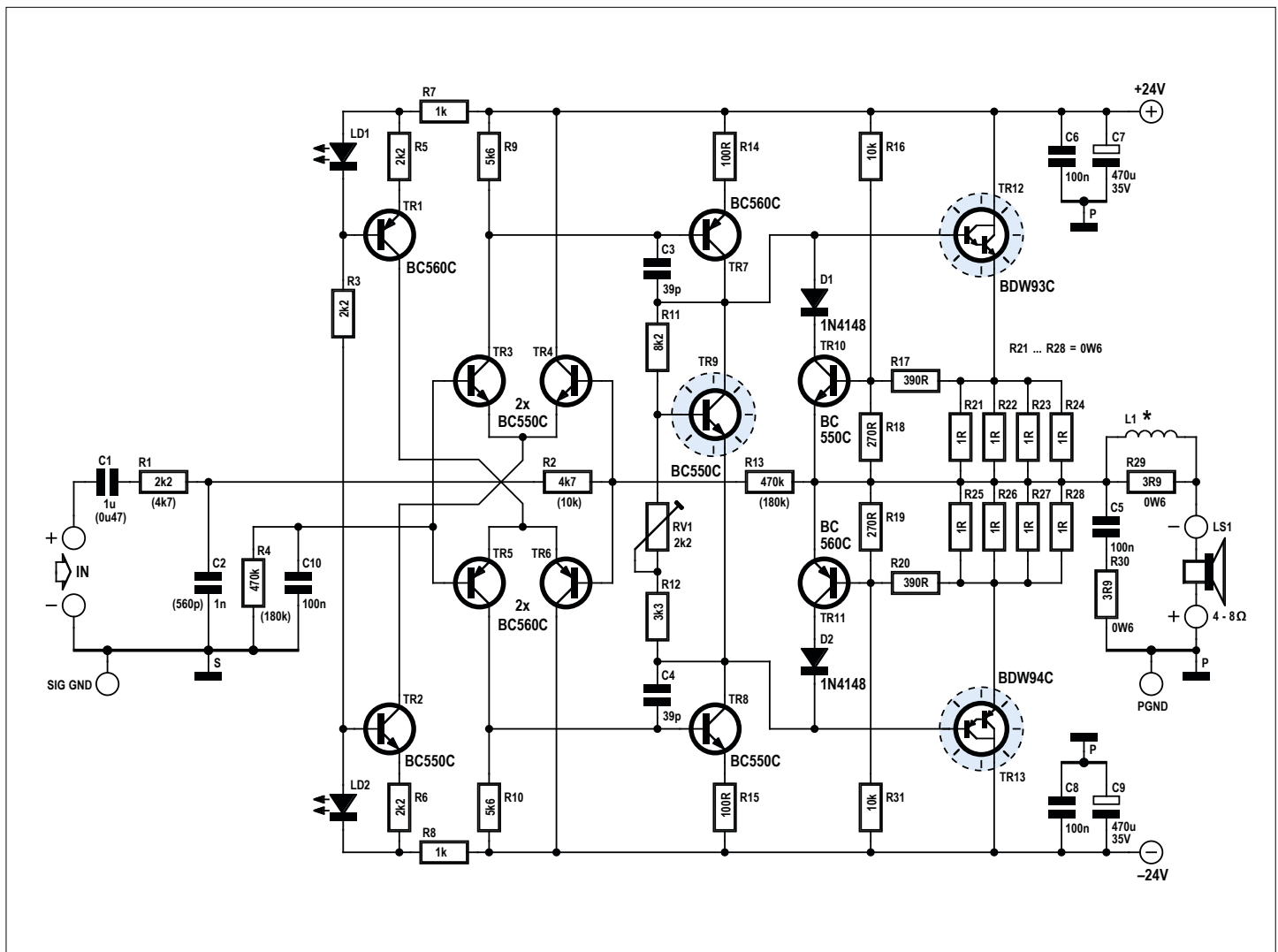


Figure 1. Schematic of the compact 60-watt audio amplifier.

and 15 years on, they are still readily available and as cheap as ever — at the time of writing, a set of output devices will cost you only £1! Those seeking ultra-low distortion figures should look elsewhere but if good quality, low cost, small size and easy construction are important to you, read on...

Into to the circuit

The circuit is shown in **Figure 1**. It's conventional in layout, a fully symmetrical design with single pole frequency compensation, but a few unusual steps have been taken to reduce component count. First, the amplifier is wired in the inverting configuration, rather than following more conventional non-inverting path. Consequently, speaker polarity must be reversed, but inverting amplifiers don't have to work as hard at correcting open-loop phase shifts as their non-inverting brethren — a definite bonus in a sim-

ple design. In addition, the input capacitor now serves double duty, providing both AC coupling and reducing DC gain to unity.

In fact, all frequency dependence is kept outside the feedback loop here, the signal being completely pre-conditioned by the existing passive input network to save a few more components.

The basic amplifier, therefore, operates with high gain to frequencies beyond the audio band and it will be noticed that aside from the 39-pF frequency compensation capacitors and some basic decoupling, the design is not festooned with mysterious RC networks 'for stability'. Despite this, the amplifier has a theoretical phase margin of 40° and the compact PCB layout keeps parasitics to a minimum — think of it as a discrete power op-amp. All measurements for this article were made with partially unshielded test leads in an electrically noisy workshop envi-

ronment, where the prototype showed no signs of instability.

In the past, most amplifiers were designed with relatively low input sensitivity (1 V_{rms}) for connection to line level sources. You can build the amplifier that way if you wish (use the component values shown in parentheses for a gain of ×12 or 22 dB), but let's face it, nowadays line-level sources are less common than low-level headphone outputs from smartphones and tablets.

How many of you have connected one of these devices to your Hi-Fi, only to find it's barely audible, even at full output? The circuit as presented has a gain of 36 dB (×67) which allows full power from a typical modern headphone output (0.2 V_{rms}) — the damping factor is adversely affected at higher frequencies but it's really most important in the lower registers anyway.

Another point of interest is the LED bias-

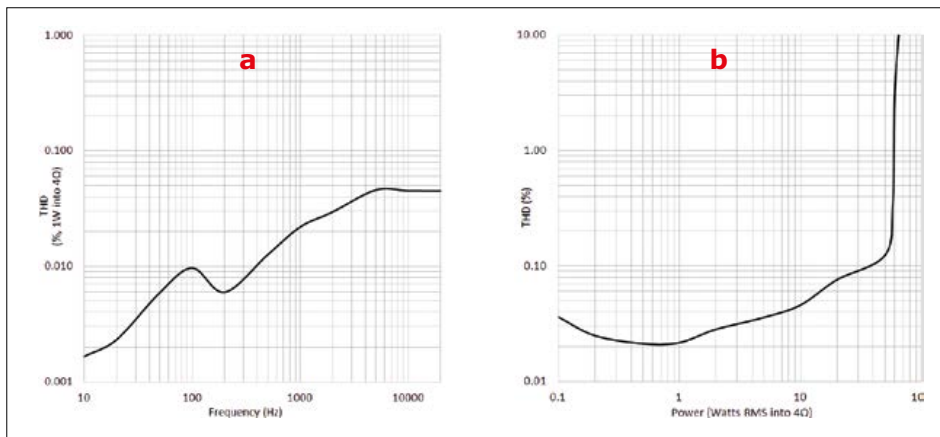


Figure 2. THD vs. frequency response (2a) and THD vs. output power response

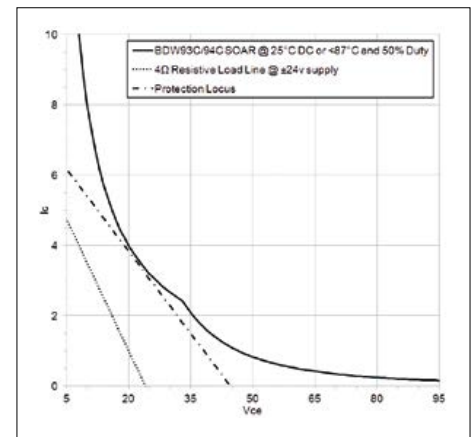


Figure 3. The protection locus is effectively determined by the amplifier loading and the power transistors' SOAR specs.

ing for the current sources — LEDs generate lower noise than zener diodes, eliminating two bypass capacitors, and they have lower thermal drift to boot! Further cost and space saving is found in the power devices' emitter resistors, where standard 1- Ω types are paralleled, giving a secondary benefit in the form of lower inductance than the conventional wire-wound choice. The output devices themselves are operated at low quiescent current, sacrificing insanely low distor-

tion figures for several practical benefits, including lower wasted energy (not just politically correct but also saves on the heatsinking) and the ability to drive the output stage with uncooled TO-92s.

Performance

Despite the less ambitious transistors used, audio quality is good, **Figures 2a and 2b** showing the usual graphs. In subjective listening tests the prototype rivalled an expensive Hi-Fi stacking sep-

arate from a well-known (high quality) manufacturer, demonstrating tight bass, accurate detail and a transparent vocal midrange.

Protection and supply

60 watts (65 watts at 10% THD) from TO-220 output transistors is driving these devices hard and so good heatsinking is important (see below). In circuit terms, this means overload protection is not considered a luxury here. Conventional single

Components List

Resistors

R1 = 2.2k Ω (4.7k Ω)
 R3,R5,R6 = 2.2k Ω
 R2 = 4.7k Ω (10k Ω)
 R4,R13 = 470k Ω (180k Ω)
 R7,R8 = 1k Ω
 R9,R10 = 5.6k Ω
 R11 = 8.2k Ω
 R12 = 3.3k Ω
 R14,R15 = 100 Ω
 R16,R31 = 10k Ω
 R17,R20 = 390 Ω
 R18,R19 = 270 Ω
 R21–R28 = 1 Ω 0.6W
 R29,R30 = 3.9 Ω 0.6W
 RV1 = 2k Ω preset, vertical
 Optional: 2 pcs 150 Ω 4W resistor for use during setup
 () = value for lower gain

Capacitors

C1 = 1 μF (0.47 μF)
 C2 = 1nF (560pF)
 C3,C4 = 39pF
 C5,C6,C8,C10 = 0.1 μF
 C7,C9 = 470 μF 35V radial
 () = value for lower gain

Inductors

L1 = 10 turns 1-mm diameter enameled copper wire, inside diameter 5mm

Semiconductors:

D1,D2 = 1N4148
 LD1, LD2 = 2mm x 5mm 2.1V, red LEDs
 TR1,TR5,TR6,TR7,TR11 = BC560C
 TR2, R3,TR4,TR8,TR9,TR10 = BC550C
 TR12 = BDW93C
 TR13 = BDW94C

Miscellaneous:

Heatsink (see text)
 Mounting kits for TR12 and TR13
 9 wire links on PCB
 PCB
 Suitable power supply, including 2A quick-blow fuses in $\pm 24\text{V}$ lines (see text for recommendations)

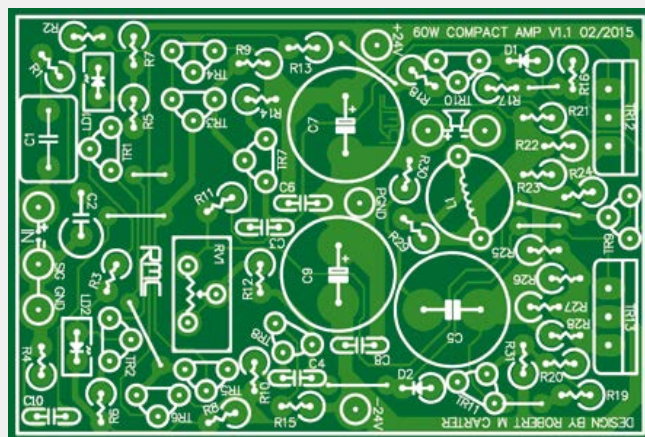


Figure 4. Printed circuit board layout shown at a scale of 150%.

slope circuits are used. The protection locus is shown in **Figure 3**, where it can be seen there is good leeway for inductive and lower than advertised resistive speaker loads (down to about $3.1\ \Omega$). The amplifier should also be protected with 2-A quick blow fuses in the supply leads. A conventional unregulated power supply is sufficient, a 100-VA 18-0-18V transformer, bridge rectifier and two 10,000- μF capacitors will supply a stereo pair. It is advisable to tin the high current PCB tracks — those related to the output transistors' collectors and emitters and the speaker load and power ground — but beyond that, construction is straightforward; aside from fitting the nine wire links, taking care to avoid solder bridges on the compact PCB (**Figure 4**, shown scaled at 150%) and ensuring that the flat side of the amplified diode (TR9) is in firm contact with the main heatsink (thermal epoxy is recommended), there is little to say.

Mind the heatsink

How large should the heatsink be? It's a subject that's often skimmed over or ignored completely, but inadequate heat-sinking is probably the number one cause of unreliability in home built amplifier designs! There's a reason why it's poorly discussed. It's because it's one of those things there's no correct answer to. Do you heatsink for sine waves or for some sort of standardized music signal? What degree of reactive load element do you consider? Under what ambient temperature do you plan to operate and what kind of thermal insulation washers will you use? Ultimately you have to make a subjective decision and either take the belt and braces approach or be practical and risk failure under unlikely conditions. During testing the output Darlington's **were** allowed to overheat and fail, the only damage being to themselves and the power supply fuses — no fused PCB tracks or cascade failures resulted and the damage was limited to about £1. The limiting factor is the maximum permissible temperature of the output devices (itself another subjective decision, unless you want to use DC dissipation values and that's like belt, braces and duct tape...). The BDW93/94 should not rise above a **case** temperature of 87°C in this design (40 W DC, higher for pulses). Under short-circuit conditions the current is limited to about 3 A by the protection

circuits (over 6 A with a speaker load) giving a total module dissipation of 72 W. Assuming a maximum ambient temperature of 40°C and good quality thermal insulating washers yielding 0.5 K/W per transistor, the heatsink will be 18°C cooler than the device cases, giving it a maximum temperature of 69°C , an increase of 29°C over ambient. That's a 0.4 K/W heatsink.

That's a big heatsink. Another approach is to use something smaller, more suited to music signals.

1.2 K/W should be sufficient. A 70°C bimetallic switch fixed to the heatsink and wired to disconnect the speaker when things get too hot would give peace of mind and possibly allow you to use something slightly smaller still. This is a **compact** amplifier after all. A fan would help too.

What about one of those CPU heatsink/fan assemblies? The maximum temperatures and power dissipations of current Intel/AMD chips are similar to what we're looking at here. The prototype was fitted to a plain aluminum heatsink of unknown rating (it **felt** right) which proved adequate for one module without a fan and suitable for a stereo pair with (gentle) forced cooling, see **Figure 5**.

One last point on cooling. The transistors' tabs are connected to their collectors and need to be electrically isolated from the heatsink. We assumed in the last paragraph the use of thermal washers giving a performance of 0.5 K/W. You'll have to be careful to achieve that. Mica washers with a thin, even coat of thermal grease on both sides are recommended; if you use the grey/pink glass/silicon variety be wary — some of them are rated for high voltage isolation rather than good thermal performance (they can be ten times worse than mica). Be sure to use bushes of the correct depth and screw the transistors down good and tight ('till a bead of thermal grease has oozed out all around) — see **Figure 6**.

Construction hints

Once construction is complete the module(s) must be wired to the power supply. To avoid hum, the center tap of the supply's capacitors should be used as the common system ground point, the PCB's power and signal ground connections being taken back to this point by separate wires (a stout wire for the power ground).

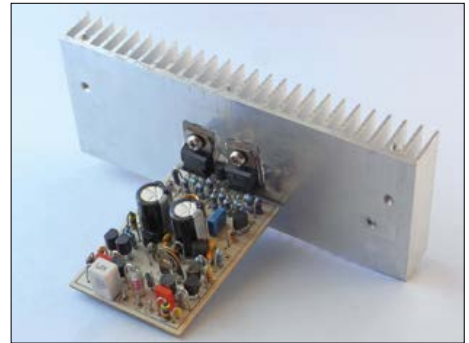


Figure 5. Do not skimp on the heatsink!



Figure 6. Low in aesthetics but definitely okay for keeping the power transistors cool.

In-line fuse holders (20-mm glass fuses were used for the prototype — if you are using more than one module, each will have to be fused separately) are a good idea for the $\pm 24\text{V}$ connections.

Before its first power-up, the prototype was checked very carefully for shorts on the PCB, the fuses temporarily replaced with 150- Ω 4-W resistors and RV1 set fully clockwise (no speaker connected!). Once switched on, the output should be close to zero volts (just a few mV offset on the prototype) and RV1 should be adjusted until the temporary 150- Ω resistors are dropping 5 V each, for a total quiescent current of 33 mA (RV1 was then at mid-travel on the prototype). Assuming all that went well the temporary resistors can be replaced with fuses, the module tested with music signals then cased up and put into service. ◀

(150183)

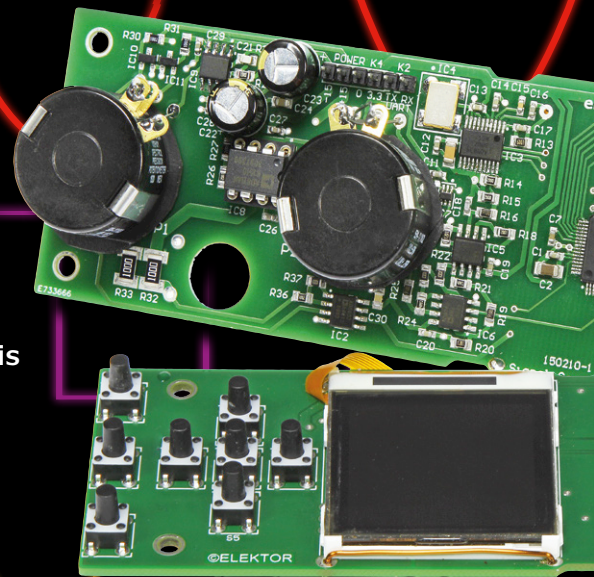
DDS Function Generator

Sines, squares and triangles up to 10 MHz

By **Theodorou Gerasimos** (Greece)

Post engineering: **Ton Giesberts** (Elektor.Labs)

DDS chips are readily available, greatly simplifying the design of the analog part of a wide-range function- or signal generator. All you need to do (they say!) is choose one, add some suitable output circuitry, pick a microcontroller, provide a user interface and start programming. To which we reply: sweet dreams, here is the real story: power to the AD9834!



A few years ago I needed a function generator for my home laboratory. In my job I had worked with some expensive commercial models and initially I had planned to buy one of these. However, none of them really was what I wanted – too complex for simple use – and so I decided to design and build my own. It turned out to be DDS (direct digital synthesis) based but that was not the only component selection issue I ran into. Here are a few more.

My components, your components?

After researching various techniques I settled on a Direct Digital Synthesis (DDS) based architecture for my project. DDS employs a digital oscillator with quartz crystal precision to accurately generate sinewaves up to very high frequencies. For the microcontroller I chose one from Analog Devices. Although well known for their Digital Signal Processor (DSP) families, this may not be the first MCU manufacturer that comes to mind. They do have a nice family though of 32-bit ARM controllers supported by something I like a lot: comprehensive documentation. Unlike other MCU manufacturers that sometimes need more than one thou-

sand pages to elucidate their devices, Analog Devices manages to fit a complete description of a complex microcontroller into a document of slightly more than a hundred pages. Analog Devices also specializes in DDS chips, operational amplifiers and other analog support chips and so made for a great one-stop shop for this project. Their friendly sampling service quickly got me started without forking out a lot of money.

The **MCU** I selected was the **ADuC7024B-STZ62**, a member of the Precision Analog Microcontroller family in a 64-pin package, containing an ARM7TDMI core running at 44-MHz clock speed. These MCUs are called analog because they feature analog inputs and outputs (yup, ADC and DAC) and an analog comparator. Besides, they sport PWM, timers and standard serial ports (SPI, UART, I²C). Our micro has 8 KB of RAM and 62 KB of flash memory that can be programmed in-circuit over a serial port. Note that some types use I²C for flash memory programming, so make sure you get the specified type and not one with a slightly different part number.

The **DDS chip** for the project is the popular **AD9834**. The maximum frequency of the external oscillator is 75 MHz, allowing

a maximum output frequency of 37.5 MHz (half the clock frequency). The downside of such a fast clock signal is a frequency resolution of just 0.28 Hz owing to the chip's 28-bit integer frequency divider. That doesn't sound like a big deal, but it does equate to almost 0.6% at 20 Hz. To improve this, the clock frequency can be lowered, albeit at the expense of the maximum achievable output frequency. At 1 MHz for instance the resolution becomes 0.004 Hz, which corresponds to an error of 0.005% at 20 Hz, but the maximum frequency is then down to a measly 500 kHz. In this project the DDS clock frequency is 75 MHz to ensure a relatively clean signal up to 10 MHz; resolution was traded against range.

The user interface is an important part of any instrument and so I spend a lot of attention on this. I added a graphic display and a bunch of pushbuttons to allow easy navigation through clear menus and options. Two multitrans potentiometers are used for quickly adjusting the output signal's amplitude and DC offset. For the **display** I used a cheap cellphone replacement display, the **Nokia 6100**, that's well-documented by the open-source community on the internet. Unfortunately



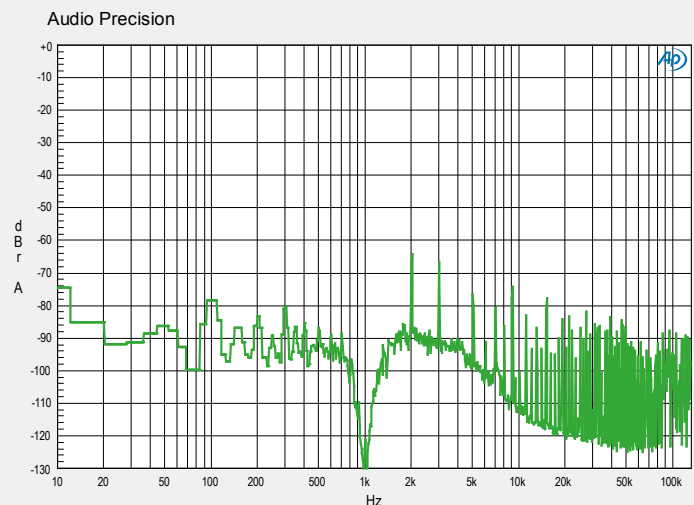
this display exists in two variants (Epson and Philips) that are not fully compatible. Even more inconveniently, in most cases you cannot specify the type you want when you buy it. To work around

this problem I wrote two versions of the software to support both types. Because looks are important too, when the electronics were completed I built the generator into a nicely milled standard

aluminum enclosure from **Hammond**. The result is a very stylish instrument that has won a preeminent place on my electronics workbench.

Specifications

- Direct Digital Synthesis (DDS) with analog front-end
- Frequency range: 1 – 10 MHz
- Frequency resolution: 0.28 Hz
- Output: 0 – 15 V_{pp}
- THD+N (100 kΩ load, B > 500 kHz):
 - 1 V, 1 kHz: 0.12% (0.09% for B = 22 kHz)
 - 5 V, 1 kHz: 0.1% (0.09% for B = 22 kHz)
 - 1 V, 10 kHz: 0.1% (0.09% for B = 80 kHz)
 - 5 V, 10 kHz: 0.09% (0.08% for B = 80 kHz)
 - 1 V, 100 kHz: 0.1%
 - 5 V, 100 kHz: 0.08%
- S/N (referred to 1 V): 72 dB
- Maximum output (10 MΩ load):
 - Sine: 16 V_{pp}
 - Triangle: 16 V_{pp}
 - Square: 18 V_{pp}
- DC offset voltage range: -10 to +10 V
- Output impedance: 50 Ω
- Duty cycle (square wave): 1 – 99%
- Rise and fall time (80%, square wave): 100 ns
- Sweep mode
- Power consumption: 3 VA



An FFT plot up to 130 kHz of a 1-V, 1-kHz sinusoidal output signal (fundamental suppressed). The THD+N is mainly caused by the harmonics of the signal. Some AC line related (50 Hz) components are visible but are close to the noise floor.

Visit www.elektor-labs.com/150210
for more figures and plots.

The circuit

Figure 1 shows the schematic of the DDS function generator. The signal generating part is at the top, the lower part shows the microcontroller and the user interface.

The passive parts around DDS chip IC3 are as recommended by the manufacturer. Its output is filtered by RC network R16/C18 before being amplified by IC5.A. The reason for such a simple filter instead

of a higher order one was simplicity. Also, as we will see later on, a high-order filter was not really necessary. The output signal from IC5.A follows two paths. The upper path is used for sinusoi-

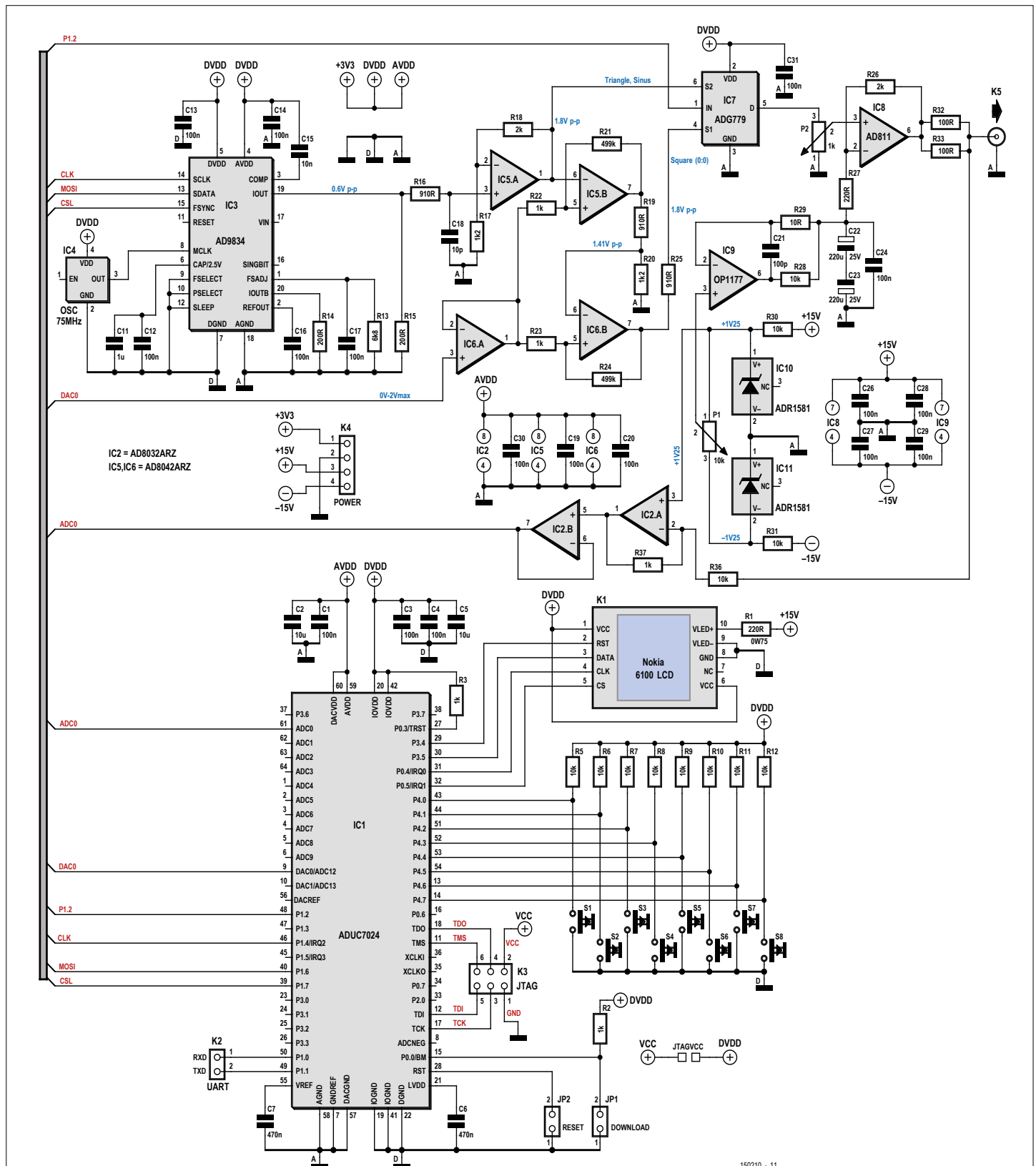


Figure 1. The signal generator's main circuit board contains everything except the power supply.

generator. To minimize power loss in the 3.3 V regulator a separate rectifier (D5/D6) with a relatively small filter capacitor is used (C17/C18). The high ripple voltage that remains reduces power loss in regulator IC3 somewhat.

Schottky diodes are used in positions D1 and D2. At maximum load the voltage drop across 'normal' diodes together with the ripple on C5 cause the input voltage of IC1 to come perilously close to its minimum value. At 0.5 A the forward volt-

age of the Schottky diodes is less than 0.45 V whereas a classic 1N4007 would drop almost twice as much.

Printed circuit boards were designed for both the main circuit and the power supply. The main board was given special attention in order to keep the output signal away from any interference produced by high-speed digital control signals. Most parts by far are SMD but assembling the board should not pose problems as long as you wear strong glasses.

The software

The source code was written in C using the well-known µVision integrated development environment from Keil. A real-time operating system (RTOS) was not used to keep things simple.

The code consists of only a few files. Some of them are header files containing definitions and function prototypes, the others contain the functions itself. Two of them are to support the two different graphic LCDs, the file 'main.c' con-

Component List Generator Board

Resistors

All 1%, 0.125 W, SMD 0805
 R1 = 220Ω, 1%, 0.75W, SMD 2010
 R2,R22,R23,R37 = 1kΩ
 R3 = 1kΩ, 1%, 0.1W, SMD 0603
 R5,R6,R7,R8,R9,R10,R11,R12,R30,R31,R36 = 10kΩ
 R13 = 6.8kΩ
 R14, R15 = 200Ω
 R16, R19, R25 = 910Ω
 R17, R20 = 1.2kΩ
 R18, R26 = 2.0kΩ
 R21, R24 = 499kΩ
 R27 = 220Ω
 R28 = 10Ω
 R29 = 10kΩ, 1%, 0.1W, SMD 0603
 R32, R33 = 100Ω, 1%, 0.75W, SMD 2010
 P1 = 10kΩ, 2W, 10-turn potentiometer
 P2 = 1kΩ, 2W, 10-turn potentiometer

Capacitors

Default: SMD 0603
 C1,C3,C4,C12,C13,C14,C16,C17,C19,C20,C26, C27,C28,C29,C31 = 100nF, 50V, X7R

C2, C5 = 10µF, 16V, X7R, SMD 1206
 C6, C7 = 470nF, 25V, X7R
 C11 = 1µF, 16V, X7R, SMD 1206
 C15 = 10nF, 50V, X7R
 C18 = 10pF, 50V, COG/NPO
 C21 = 100pF, 100V, COG/NPO, SMD 0805
 C22, C23 = 220µF, 25V, radial, 3.5mm pitch, diam. 8mm max.
 C24, C30 = 100nF, 50V, X7R, SMD 0805

Semiconductors

IC1 = ADUC7024BSTZ62, LQFP-64, programmed*
 IC2 = AD8032ARZ, SOIC-8
 IC3 = AD9834BRUZ, TSSOP-20
 IC4 = FXO-HC736R-75, 7 x 5 mm
 IC5, IC6 = AD8042ARZ, SOIC-8
 IC7 = ADG779BKSG-REEL7, 6-Lead SC-70
 IC8 = AD811ANZ, DIP-8
 IC9 = OP1177ARZ, SOIC-8
 IC10, IC11 = ADR1581ARTZ-REEL7, SOT-23-3

Other

K1 = Socket, 0.5 mm, 1.5 mm stack, 10-way,

DF23C-10DS-0.5V(51), Hirose (HRS)
 K2, JP1, JP2 = 1x2 pin header, vertical, 0.1" pitch
 K3 = 2x3 pin header, vertical, 0.1" pitch
 K4 = 1x4 pin header, vertical, 0.1" pitch
 K5 = BNC 50 Ω, Straight Bulkhead Jack, Panel mount
 S1-S8 = 6 mm tactile switch, actuator length 4.9mm, 24V/0.05A, SPST-NO
 8-way DIP socket for IC8,
 Graphic B/W replacement LCD for Nokia 6100

Miscellaneous

Aluminum enclosure, Hammond type 1455T1601, 165 x 160 x 51.5mm
 Optional: EMI/EMC filter, inlet, IEC, 250VAC / 4A
 2 knobs, black, 16mm, 0.25 in. shaft diam.
 PCB 150210-1 v1.11 (www.elektor.com)

* Not available ready programmed from www.elektor.com

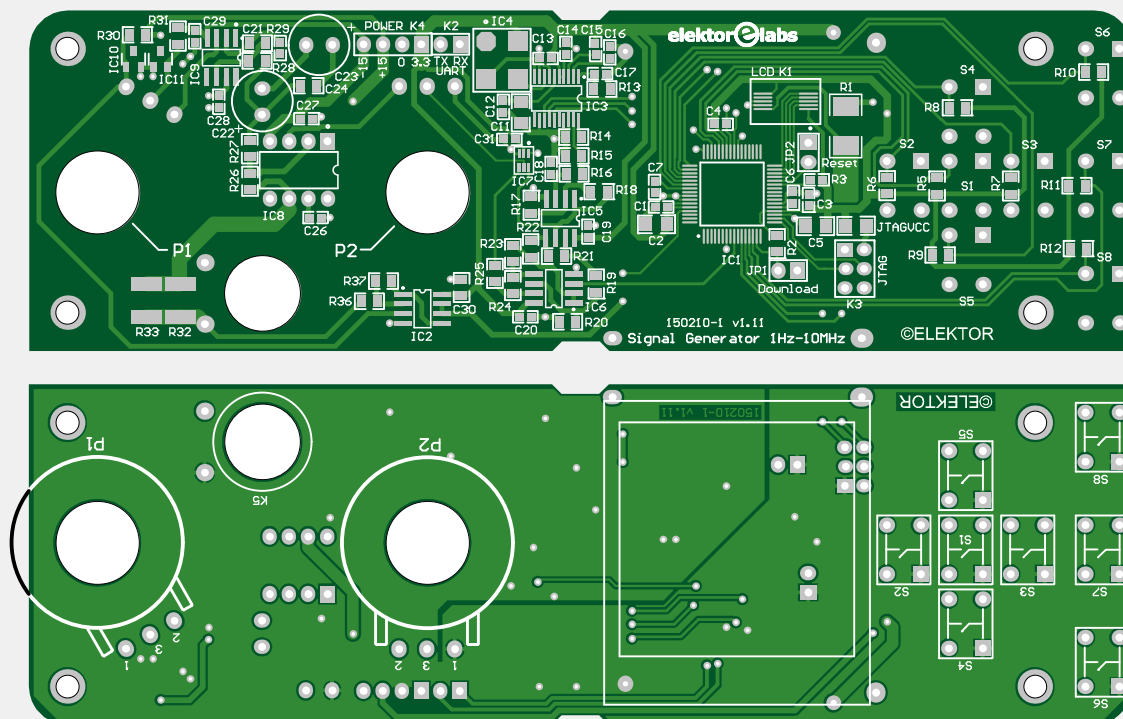


Figure 4a. Component overlays of the generator PCB (double-sided SMD populated).

tains all the signal generator code. The file 'init.s' is written in assembly language and contains the functions to initialize the microcontroller. This file was written from scratch and is not part of the development environment.

The LCD driver is partly based on open source code found on the Internet and partly written by myself. Since the display is not connected to a hardware SPI port, the communication protocol is emulated and bit-banged on GPIO pins. The functions WriteLcdCommand and WriteLcdData take care of this. Most of the display driver coding effort went in creating two fonts, a large and a small one.

The keyboard is handled by polling in the main endless loop.

To keep code size small and IDE-independent, an attempt was made to avoid using floating-point arithmetic and math libraries.

All source files have been compiled into so-called ARM thumb code (16-bit code). Not so much reducing the size of the hex file, but because the flash memory of the microcontroller is 16-bit wide, making 16-bit code faster to execute. The total size of the executable is about 8 KB plus another 20 KB to hold the splash screen

that appears at power up. No tricks were needed to fit all this in the MCU because its flash memory is large enough for the purpose.

The software archive for the project can be downloaded FOC from [1].

Building it

Two PCBs have been designed to hold all the parts; they are printed in **Figure 4** along with the component lists. As mentioned, assembling the main PCB is not

too difficult as long as you have good eyes and/or a magnifying glass. Note that the pushbuttons and the LCD must be mounted on the bottom side of the PCB. The LCD requires some special attention as its flexible flat cable must be detached from its plastic support (**Figure 5**) so that it can be folded over the PCB and reach K1 (**Figure 6**).

The two potentiometers should stick through the PCB with the shaft protruding from the bottom side. At Elektor.Labs

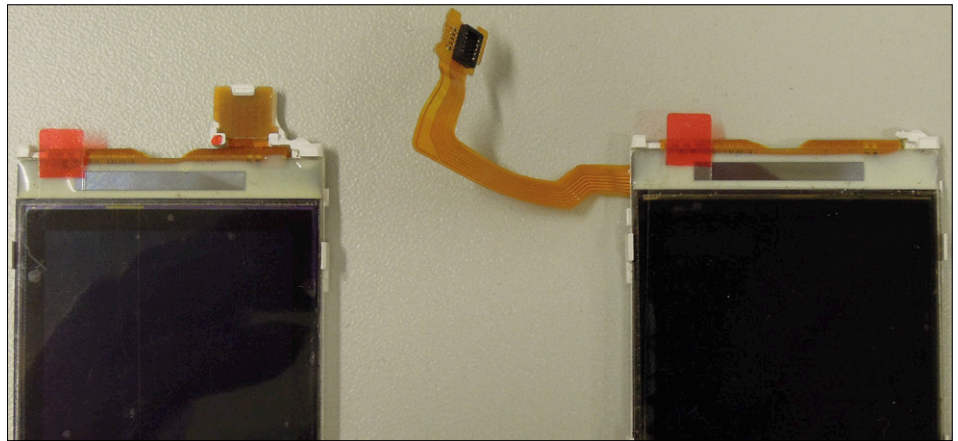


Figure 5. On the left, the display as used in some mobile phones of a well-known Scandinavian brand (ain't Ikea), on the right the same display adapted for our generator.

Component List Power Supply

Resistors (1%, 0.6 W)

R1, R3 = 180Ω
R2, R4 = 2.0kΩ
R5 = 240Ω
R6 = 390Ω
R7 = 10kΩ, 5%, 0.25W

Capacitors

C1, C2, C3, C4, C15, C16 = 10nF 50V, Y5V, 0.2" pitch
C5 = 1000μF 50V, 5 or 7.5 mm pitch, 16mm diam.
C6 = 470μF 50V, 5 or 7.5mm pitch, 13mm diam.
C7, C8, C13, C14, C18, C21 = 100nF 50V, X7R, 0.2" pitch
C9, C10, C11, C12, C19, C20 = 10μF 50V, 2mm pitch, 6.3mm diam. max.
C17 = 47μF 50V, 2.5mm or 3.5mm pitch, 8mm diam. max.

Semiconductors

D1, D2 = STPS2L60, DO-41 case
D3, D4, D5, D6 = 1N4007, DO-41 case
IC1, IC3 = LM317, TO-220 case
IC2 = LM337, TO-220 case
LED1 = LED, green, 3mm

Miscellaneous

K1 = 2-way PCB screw terminal block, 0.3" pitch, 500V

K2 = 4-way (2x2) PCB screw terminal block, 0.2" pitch, 250V
TR1 = 2x115V prim./2x15V sec., 10VA, e.g. Block FL 10/15
F1 = fuse, 100 mA (230 VAC line) or 200mA (115 VAC line); slow blow, 250V, 20 x 5mm
F2, F3 = fuse, 315mA, slow blow, 250V, 20x5mm

Fuse holder for F1, F2, F3, 20 x 5mm, 500V, 10A
Covers for F1, F2, F3 fuse holders, 20 x 5mm
JP1 = Jumper wire
PCB 150210-2 v1.1 (www.elektor.com)

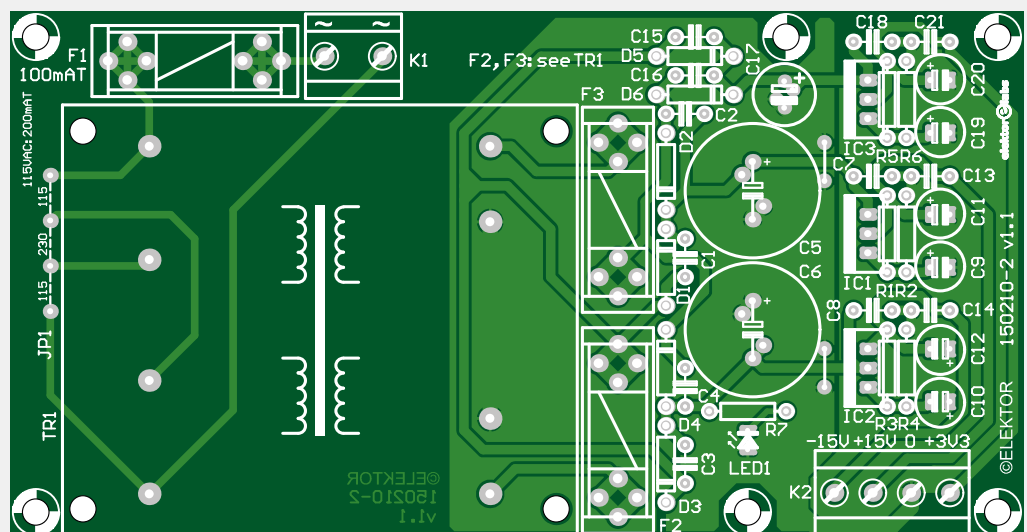


Figure 4b. Component overlay of the power supply PCB (single-sided TH populated).

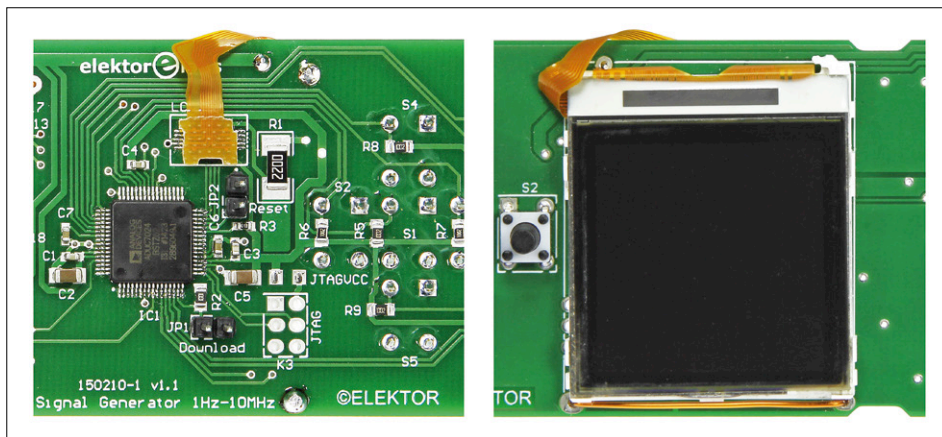


Figure 6. This is how the display is supposed to be mounted. Use double-sided tape to stick the display to the board.

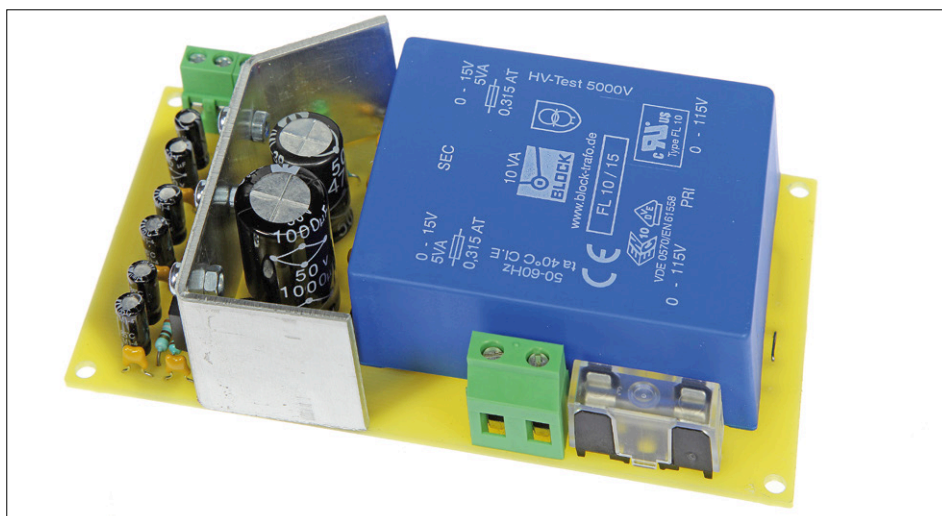


Figure 7. The assembled power supply including the DIY heatsink. Note the wire jumper behind the transformer to select the line voltage (230 VAC in this case).



Figure 8. The prototype built at Elektor.Labs by Jan Visser. All cutouts were done with a 100- μ CNC machine simple tools, care & patience.

we used homebrew rubber 'disks' (cut from an old bicycle inner tire) to prevent the potentiometers from slipping while adjusting them. 10 mm for the disk's hole diameter is fine, and the outside diameter is preferably a little under 22 mm. In the case of P2 the rubber disk also prevents damage to a copper track on the top side next to the hole for P2.

The power supply PCB should not pose any problems. We went for a transformer with two primary windings to support both 115 VAC and 230 VAC grid voltages. A jumper wire has to in place at JP1 (the middle one of three dotted lines) if you are on 230 VAC. Two jumper wires are necessary for 115 VAC (the two outer dotted lines). Do not install all three jumper wires! Also don't forget to fit the correct primary fuse: 100 mA(T) for 230 VAC and 200 mA(T) for 115 VAC, where (T) is time delay.

The three regulators require a heatsink which we made out of a single 2-mm thick aluminum strip (**Figure 7**). Don't forget the electrical insulation (mica washer and plastic bush) for the three regulators. The M3 screw should be about 6 mm long. A metal washer between the head of the screw and the plastic bush is advised. Often the plastic bush is a bit too long so cut it to the proper length (with a hobby knife) before mounting the regulators on the heatsink.

If you decide to use the same enclosure as we did (**Figure 8**) then you can download a mechanical drawing from the Elektor website [1] showing milling details of the front and back panels.

BNC socket K5 is isolated and should be mounted on the front panel. After fixing the main PCB to the front panel (we glued the screws to the back of the panel), connect the BNC with short wires to the PCB.

Programming

Before the signal generator will work it must be programmed with the right firmware. Because of the two possible LCD configurations, two pieces of firmware are available from our website [1]. Since you can't tell the exact type of the display by looking at it, it is a matter of trial and error to find the right software. Nota dicky bird from the function generator after programming the MCU? Try the other firmware. No wave either? Uh-oh... There are two ways of programming the MCU: JTAG or bootloader.

The first option requires a JTAG adapter

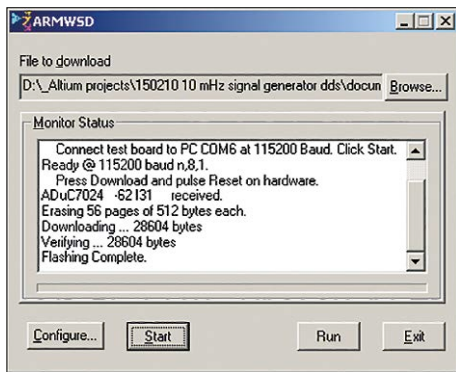


Figure 9. Screenshot showing the firmware programming tool ARMWSD in action.

and also gives you a debugging interface. Well known JTAG adapters are the J-LINK from Segger and Keil's ULINK. The standard JTAG interface has 20 pins, but it can work with only six pins too (K3). The second option is over the serial port interface (K2). This is a programming-only option, debugging is not possible over this interface. A suitable cable

ELEKTOR STORE PRODUCTS

- 150210-1 – Main board, unpopulated
- 150210-2 – Power supply board, unpopulated
- 150210-91 – Main board, ready populated

We regret we cannot supply microcontroller IC1 blank or ready-programmed

can be purchased from Analog Devices but it is very easy to make your own with a serial TTL-to-USB cable.

A software serial programming tool (ARMWSD.exe) is available for free from the Analog Devices website. First select the appropriate USB serial port and load the hex file. The program will then ask you to "Press Download and pulse Reset on hardware" (Figure 9). That's why the two jumpers JP1 and JP2 landed on the PCB labeled 'Download' and 'Reset' respectively. If a jumper is fitted on Download at power-on, the microcontroller will stay in bootloader mode and the display will remain dark, so don't forget to remove it after programming.

There you are

Now that you have built and tested this nice DDS function generator you no longer have any excuse for refusing amplifier repair jobs. Graphing a filter's transfer curve has become child's play. Riding the waves will come natural to you. Welcome to the wonderful world of well-equipped electronics engineering (WWW-EEE)! ◀

(150210-I)

Web Links

- [1] www.elektormagazine.com/150210
- [2] www.elektor-labs.com/150210

User Manual

PB	Function	PB	Function
S1	Set	S5	-Down
S2	Right	S6	Mode
S3	Left	S7	Sweep
S4	+Up	S8	Calibration

- **Waveform** – Press *Mode* to toggle between sine, square and triangle.
- **Duty Cycle** – The duty cycle can only be set in square wave mode. Press *Mode* to activate the square wave output. The duty cycle is indicated on the bottom line of the display. Adjust the duty cycle by pressing *+Up* and *-Down* (the digits must not blink).
- **Frequency** – Press *Set*. A digit starts blinking. Use *+Up* and *-Down* to change the value of the blinking digit; use *Left* and *Right* to navigate through the digits. When done press *Set*.
- **Amplitude** – Adjust P2. Note that adjusting the amplitude affects the offset voltage. See [2] for detailed measurements.
- **DC offset** – Adjust P1.
- **Frequency sweep** – Press *Sweep* to open the sweep menu. The least significant digit of the start frequency is blinking. Use *+Up* and *-Down* to change the value of the blinking digit; use *Left* and *Right* to navigate through the digits. When done press *Set* to advance to the next parameter. Set the stop frequency, the sweep time (called "msec") and the sweep mode (logarithmic or linear). Press *Set* to start the sweep. This is indicated by "sweep run" in the first line of the sweep menu. Press

Set again to stop the sweep ("sweep stop" is displayed in the first line) and new values can be set. Press *Sweep* to return to the main menu.

- **Contrast** – Press *Calibration* to open the calibration menu where the LCD contrast can be set. Use *Set* to navigate to the contrast option, then use *+Up* and *-Down* to change the contrast level. Press *Calibration* to return to the main menu.
- **Calibrate voltage levels** – Connect an oscilloscope to the generator's output and set the output level to $5 V_{pp}$. Press *Calibration* to open the calibration menu. Select *Measurements* to start the calibration procedure. (If entered by mistake, the only way to get out is by switching off the power.) Adjust P1 to set the minimum value of the output signal to 0.00 V, press *Set* when done. Adjust P1 to change the maximum value of the output signal to 12.00 V. Press *Set* when done. A message appears to indicate that calibration has been completed. Press *Calibration* to return to the main menu.
- **Calibrate frequency** – Connect a high-precision frequency counter to the function generator's output. Press *Calibration* to open the calibration menu. Select *Frequency* to start the calibration procedure. (If entered by mistake, the only way to get out is by switching off the power.) Adjust the output frequency to 100,000 Hz by pressing *+Up* and *-Down*. Press *Set* when done. A message appears to indicate that calibration has been completed. Press *Calibration* to return to the main menu.

BL600 e-BoB

Part 5

Bluetooth Low Energy Module

- SPI port & digital/analog converter
- Android application

By **Jennifer Aubinais** (France) elektor@aubinais.net

Following on from the BL600's I²C port we looked at last time, we're going to take a look here at its serial interface. To do this, we're going to be using a digital/analog converter, itself fitted with an SPI port. This will be an opportunity to produce your own Bluetooth application.

Out of the BL600's impressive arsenal, we're going to take a look here at its serial interface, referred to as SPI. To do this, we're going to be using a 16-bit digital/analog converter (DAC), itself fitted with an SPI port. The joint information will be an opportunity to produce your own Bluetooth application. Obviously, it's not our aim to give a course in development under Android, but the article gives all the elements you need for your phone and the BL600 to be able to get along well together. If there is enough demand, perhaps one day we'll present here an Android application devoted entirely to our SPI device. Note: the tools and commands used in this article have already been presented in the previous articles in this series. [1].

SPI port and DAC

The LTC1655L DAC from Linear Technology [2] can be powered at 3.3 V, and if we use the internal voltage reference, it operates with no other components (**Figure 1**). In this case, its output voltage is limited to 0–2.5 V. The value we want this output voltage to take will be sent to the DAC's serial port by our BL600 e-BoB's SPI port (pin 10, MOSI = Master Output,

Slave Input and pin 12, CLK (**Figure 2**). The DAC also has a serial output that we've connected to the input of the e-BoB's SPI port (pin 11, MISO = Master Input, Slave Output), which will let us check our circuit is working. Using a voltmeter connected to the DAC output (pin 7), we'll measure its output voltage and adjust the drive so as to obtain a voltage of e.g. 1 V.

Our BL600 module and the DAC are now connected. Now let's take a look at the BL600's SmartBASIC program. Here's what happens in the *LTC1655L.sb* program (**Listing 1**), downloadable from the Elektor Magazine website [4]:

a) Open SPI

The first step is to open the SPI port and handle its return code using a simple *if* condition.

```
Rc = SpiOpen(0,1000000,0,handle)
```

- **nMode = 0**: CPOL (*Clock Polarity*) to 0 and CPHA (*Clock Phase*) to 0

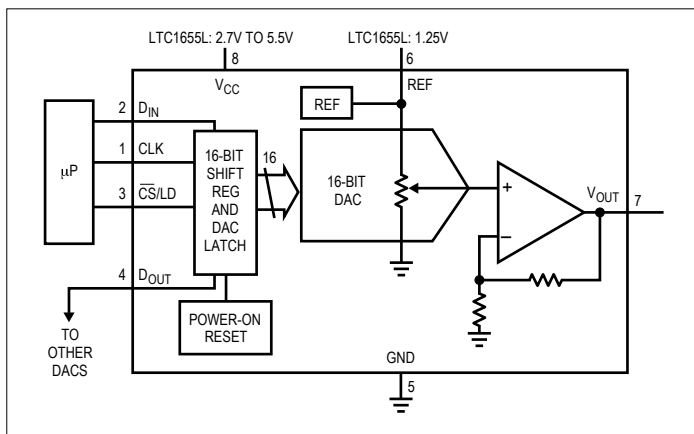


Figure 1. Internal circuit of the LTC1655L analog/digital converter (ADC) used to illustrate communication via the SPI interface.

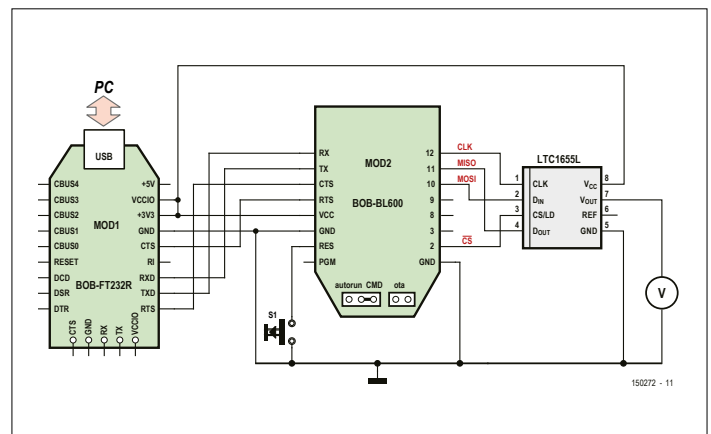


Figure 2. Circuit of the BL600 using the LTC1655L. The use of the FT232 e-BoB was described in the previous articles.

- **nClockHz** = 1000000: clock frequency
 - **nCfgFlags** = 0: must be set to 0.
 - **nHandle** = if the return code is 0, this variable can then be used to read, write, and close the SPI interface.
- rc** = the return code must be zero to be able to continue. If this is not the case, we convert its value into a character string via the *SPRINT* function using the option *INTEGER.H* and display it. Then we stop the program (*STOP*).

b) Writing and reading the SPI bytes

The aim is to obtain 1 V at the DAC output. We shall also read the data we've sent – this will be a good exercise.

• Calculation for an output voltage of 1 V

The DAC's output voltage range of 0–2.5 V is covered using 16-bit words (2^{16}) from 0 to 65,536. Hence its resolution is $2.5/65,536 \approx 38 \mu\text{V}$. Thus the value to be sent in order to obtain 1 V will be $(1/2.5) \times 65,536 \approx 26,214$ (= 6666 in hexadecimal, written as 0x6666).

• Writing the two bytes

As the function for sending data via the SPI port only accepts strings of ASCII characters, the digital values to be sent will need to be converted. Now the DAC expects a 16-bit word, so we shall have to convert the digital value 26,214 into two hexadecimal values that will form the character string 0x66-0x66, i.e. *ff* in ASCII. By sending this string, we obtain a voltage of 1 V at the DAC output.

Rc = SpiReadWrite(stWrite\$,stRead\$)

- **stWrite\$** = character string of the data to be written
- **stRead\$** = character string of the read data, the same length as the **stWrite\$** character string.
- **rc** = The return code must be 0..

• Reading from the SPI port

The SPI port is synchronous (full duplex), we can write to it and read from it at the same time. Our function to write data to the SPI port is also a read function. And since the DAC's serial output is looped to the e-BoB's MISO input and the DAC sends back the received data, we receive an echo of the data sent during the previous operation to write to our DAC.

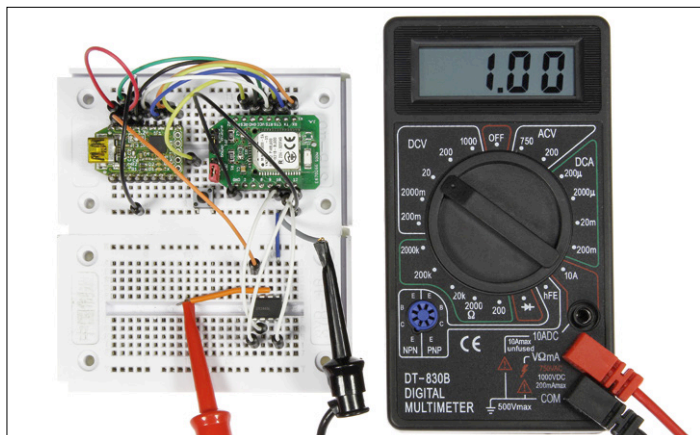


Figure 3. Photo of the experimental assembly on a prototyping board.

Component List

Semiconductors

IC1 = LTC1655L

Miscellaneous

S1 = pushbutton

MOD1 = FT232 e-BoB, ready assembled, # 110553-91 [7]

MOD2 = BL600 e-BoB, ready assembled, # 140270-91 [7]

Or printed circuit board # 140270-1 [7]

c) Close SPI

To finish, we close the SPI port

SpiClose(nHandle)

- **nHandle** = value created by SPIOpen

Android application

The moment has come to set about programming an application under Android. On the BL600 manufacturer's website [3], you will find the source code for the *Toolkit* application which includes the following services:

Listing 1

```
//-----
DIM rc
DIM handle
DIM stWrite$, stRead$
//-----
rc = GpioSetFunc(2,2,0) // pin 2 at low
//-----
rc=SpiOpen(0,1000000,0,handle)
if rc!= 0 then
print "\nFailed to open SPI with error code
";integer.h' rc
else
print "\nSPI open success"
endif
//-----
GpioWrite(2,0) // pin 2 at low
stWrite$ = "\66\66" : stRead$=""
rc = SpiReadWrite(stWrite$, stRead$)
if rc!= 0 then
print "\nFailed to ReadWrite"
else
print "\nWrite = 0x";strhexize$(stWrite$)
endif
//-----
stRead$=""
rc = SpiReadWrite(stWrite$, stRead$)
if rc!= 0 then
print "\nFailed to ReadWrite"
else
print "\nRead = 0x";strhexize$(stRead$)
endif
//-----
GpioWrite(2,1) // pin 2 at high
SpiClose(handle) //close the port
SpiClose(handle) //no harm done doing it again
print "\nCLOSE SPI\n"
//-----
```

```

Terminal | BASIC | Config | About
CTS DSR DCD RI RTS DTR BREAK LocalEcho LineMode Clear ClosePort

AT+FWRH "01000110CF3000009001CF3001009001FA3073000002E92200000110EF2200"
AT+FWRH "00D2300000000BC14F634450162010110CE211000FB000C0014000A466169"
AT+FWRH "6C656420746F20526561645772697465CC213101A52000000110AF202D0101"
AT+FWRH "10CE211000FB0007000A000A52656164203D203078CC214801A5200000CE21"
AT+FWRH "1000CF3001009001FA301C000001CC10CA210100A520000001100110D23000"
AT+FWRH "000200D23000000100F930120004000110EF220200F930110002000110EF22"
AT+FWRH "0200F930110002000110CE211000FB0008000B000A434C4F5345205350490A"
AT+FWRH "00CC217B01A52000000110FD10F510"
AT+FCL

AT+RUN "LTC1655L"

SPI open success
Write = 0x6666
Read = 0x6666
CLOSE SPI

00

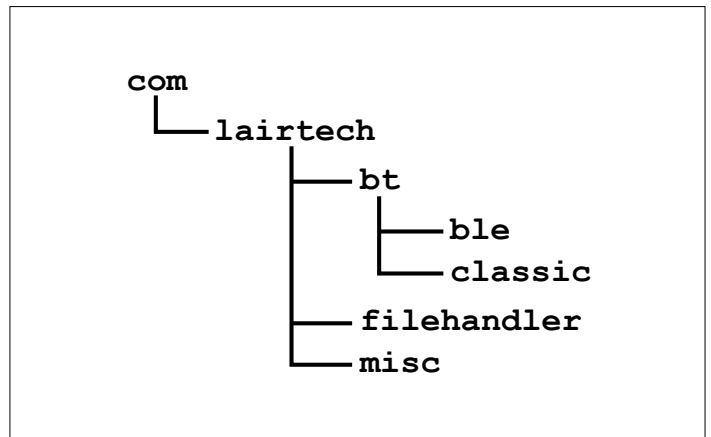
```

Figure 4. If the voltmeter displays a voltage of 1 V, the LTC1655L.sb program has run successfully.

- BPM (blood pressure)
- HRM (heart rate)
- *Proximity*
- HTM (medical thermometer) used with the temperature sensor in the previous article.
- Serial (UART)
- OTA (Over The Air)
- Batch

We are offering you for download [4] the extremely simplified source code for an application using just the UART service. We have used *Android Studio*, available under Windows, MAC OS and Linux [5]

The manufacturer has created a library *laird_library_ver.0.18.1.1.jar* in order to speed up development of Android applications in normal Bluetooth and Bluetooth Low Energy.



The tree structure in the library.

Documentation is available with the library download. There are three classes:

- BluetoothAdapterWrapper Class
- Initialization of the Bluetooth
- Verifies if Bluetooth is present
- Detects Bluetooth peripherals
- BluetoothAdapterWrapperCallback Interface
- Detects, stops Bluetooth devices
- Handles Call-Backs
- BleDeviceBase Abstract Class
- Access to the Bluetooth methods for initializing the connection to the peripheral. For example: connect, isConnected etc.

In our program, you'll find the tree structure in the library.

Listing 2

<LinearLayout

```

    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_below="@+id/textView"
    android:layout_alignParentStart="true"
    android:layout_above="@+id/btnSend">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/scrollViewVspOut"
        android:background="#ffffff">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:id="@+id/valueVspOutTv" />

    </ScrollView>
</LinearLayout>

```

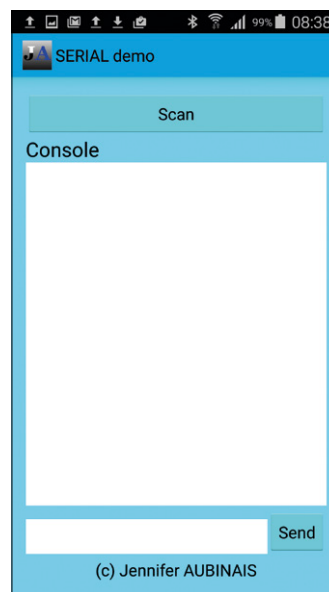


Figure 5. As simple as possible: the screen for our Serial application.



Figure 6. Display of the *mDialogFoundDevices Dialog* class with two peripherals.

▶ To thrive, connected object networks need low power consumption wireless communication. This breakout board for the Bluetooth BLE module is the dream accessory for exploring the IoT.

The program

The complete program can be downloaded from the Elektor site [4]. The first step is to create your project. As *Package name* we've chosen: `com.ja.serial` (where `ja` are the author's initials)

Screen layout: To start off, let's keep it simple: two buttons *btnScan* and *btnSend*, an area for displaying the received data using scrolling *scrollViewVspOut* and *valueVspOutTv*, and the text entry box *valueVspInputEt*. And that's all (**Figure 5**).

Example of *activity_main.xml* for the received data display area with scrolling (**Listing 2**).

Permissions: The *AndroidManifest.xml* file allows our program to access the Bluetooth. If you forget this, it will cause an error in the application that is difficult to identify.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

Declaration in the program: We're going to import different *classes* with no changes to the original sources into the following directories:

- `com.ja.bt.ble.vsp`
- `FifoAndVspManager.java`
- `FifoAndVspUiBindingCallback.java`
- `FileAndFifoAndVspManager.java`
- `VirtualSerialPortDevice.java`
- `VirtualSerialPortDeviceCallback.java`
- `com.ja.serial.bases`
- `BaseActivityUiCallback.java`
- `com.ja.serial.serialdevice`
- `SerialManager.java`
- `SerialManagerUiCallback.java`
- `com.ja.serial`
- `ListFoundDevicesHandler.java`

Importing classes from the manufacturer's library is described in **Listing 3**.

Given that the two classes *SerialManagerUiCallback* and *BluetoothAdapterWrapperCallback* are implemented, you'll find all their classes specified in our program. These can be recognized by the presence of comments at the start (**Listing 4**).

The Dialog class appears when scanning for peripherals. Then, from the list, the application connects to our BL600 e-BoB. Watch out: the application only works with peripherals that have the UART service.

The buttons: We find the *setOnClickListener* for our applica-

tion's two buttons: *btnScan* and *btnSend*. Here's what happens for *btnSend*: the text to be sent is read in the text box; if it is other than null, it is copied into the receive text area then sent using the *startDataTransfer* function (**Listing 5**).

(Tip) Launching the connection automatically: To establish an automatic connection, let's remember how we do it manually: we start the scan using the *btnScan* button, then we select our peripheral in the *Dialog* box. So for our automatic connection, we leave the main program to execute the code for the *setOnClickListener* function, then, instead of displaying the list of peripherals in *Dialog*, we start the connection to the module entered in the code *name*.

(Tip) Receiving and sending data: To enter the data to be sent and display the data received, we have two text boxes: *valueVspInputEt* and *valueVspOutTv*. There's nothing to stop your program sending the data via some other action. For example, you can send 0 or 1 depending on the position of a button *Switch*. You can handle the characters received in your program as we have done in the Thermometer application described in the January/February 2015 issue [6]. There, we performed a complex calculation that couldn't be done by the BL600 before displaying the result in large characters.

The DAC in Bluetooth

You now know how to create a Bluetooth application for the BL600 e-BoB. All you have to do is take as your base the program *upass.vsp.sb* from the firmware bundle downloadable from the manufacturer's website [3], rename it to *\$autorun\$.LTC1655L.uart.sb* [4], then create the handler *HandlerLoop*

Listing 3

```
import com.lairdtech.bt.BluetoothAdapterWrapperCallback;
import com.lairdtech.bt.BluetoothAdapterWrapper;
import com.lairdtech.bt.ble.BleDeviceBase;
import com.lairdtech.misc.DebugWrapper;
```

Listing 4

```
/*
 * *****
 * SerialManagerUiCallback
 * *****
 */
/*
 * *****
 * Bluetooth adapter callbacks
 * *****
 */
```

Listing 5

```

mBtnSend.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        switch(v.getId())
        {
            case R.id.btnSend:
            {
                String data = mValueVspInputEt.getText().toString();
                if(data != null){
                    mBtnSend.setEnabled(false);
                    mValueVspInputEt.setEnabled(false);
                    if(mValueVspOutputTv.getText().length() <= 0){
                        mValueVspOutputTv.append(">");
                    } else{
                        mValueVspOutputTv.append("\n\n>");
                    }

                    mSerialManager.startDataTransfer(data + "\r");

                    InputMethodManager inputManager = (InputMethodManager)
                        getSystemService(Context.INPUT_METHOD_SERVICE);

                    inputManager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(),
                        InputMethodManager.HIDE_NOT_ALWAYS);

                    if(isPrefClearTextAfterSending == true){
                        mValueVspInputEt.setText("");
                    } else{
                        // do not clear the text from the editText
                    }
                }
                break;
            }
        }
    }
});

```

(renamed to *MyHandlerLoop*) in order to recover the data received in Bluetooth (**Listing 6**).

Here is the list of the various actions to be carried out by the *handler* in order to process the character string received via Bluetooth:

- receive via Bluetooth the text corresponding to the desired voltage;
- convert the received text into a value to be sent to the DAC;
- send the value to the SPI port;
- receive a value via the SPI port;
- convert the value received into a character string;
- send the string via Bluetooth.

In red: you've already seen this code. We store the characters received via Bluetooth in the variable *text\$*. If the character string contains the return code 0x0D (end of string), we enter the *IF* condition.

In green: First of all, we recover the value of the voltage wanted at the DAC output. To do this, we convert the character string into a numeric variable using the *StrValDec* function, then we calculate the value for the DAC. Watch out: if the value received yields a result higher than 65,535 ($2^{16}-1$), we force the value to 65,535, a 16-bit value. In order to adhere to the format of the *SpiReadWrite* function, the value is converted into two characters using the *StrSetChr* function, thereby creating a character string *stWrite\$*.

Listing 6

OnEvent	EVVSPRX	call MyHandlerLoop //EVVSPRX is thrown when VSP is open and data has arrived
OnEvent	EVUARTRX	call MyHandlerLoop //EVUARTRX = data has arrived at the UART interface
OnEvent	EVVSPTXEMPTY	call MyHandlerLoop
OnEvent	EVUARTTXEMPTY	call MyHandlerLoop

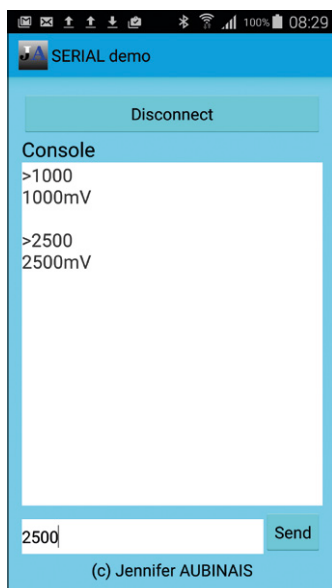


Figure 7. Using our Android application to control the output voltage of the LTC1655L DAC.

In blue: We shan't be coming back to this part of using the SPI port. The function *SpiReadWrite* is executed twice in order to recover the DAC value written in the first *SpiReadWrite*. The value written cannot be read from the DAC output port, so it will be necessary to write again so as to read the previously-loaded value. This is an oddity of the SPI protocol rather than of the device itself.

In orange: Now the value read from the DAC is a character string. We need to extract the two numerical values (high value and low value) via the *StrGetChr* function. Then we calculate the voltage value in millivolts using an inverse calculation. The conversion of

the numerical value into a character string is achieved using the *SPRINT* function. To end, we transmit the voltage value to the Bluetooth via the *BleVspWrite* function (**Figure 7**). ❏

(150272)

Web Links

[1] Previously in the series of articles published on the BL600:

1. BL600-eBoB (1): Bluetooth Low Energy communication module: www.elektormagazine.com/140270
2. BL600-eBoB (2): Editing, compiling, and transferring a program using the BLE module www.elektormagazine.com/150014
3. BL600-eBoB (3): smartBASIC programming for the Bluetooth Low Energy module www.elektormagazine.com/150129
4. BL600-eBoB (4): The I²C port and the temperature sensor: www.elektormagazine.com/150130

[2] www.linear.com/product/LTC1655

[3] https://laird-ews-support.desk.com/?b_id=1945

[4] Elektor July & August 2015, www.elektormagazine.com/150272

[5] <https://developer.android.com/sdk/index.html>

[6] Wireless thermometer: Elektor January & February 2015, www.elektormagazine.com/140190

[7] e-BoB BL600 www.elektor.de/bl600-e-bob-140270-91
e-BoB FT232 www.elektor.de/ft232r-usb-serial-bridge-bob-110553-91

Listing 7

```
function MyHandlerLoop()
  DIM n, rc, tempo$, tx$, text$
  DIM value, valtemp, pos, return$
  DIM handle
  DIM stWrite$, stRead$
  // Wait return from received data
  tx$ = "0D"
  return$ = StrDehexize$(tx$)
  tempo$ = ""
  n = BleVspRead(tempo$,20)
  text$ = text$ + tempo$
  pos = STRPOS(text$,return$,0)
  IF ( pos >= 0 ) THEN
    //-----
    // convert Voltage value for LTC1655
    //-----
    value = StrValDec(text$)
    value = value * 65535
    value = value / 2500
    IF value > 65535 THEN : value = 65535 : ENDIF
    // Init string for SPI write
    stWrite$ = "00"
    valtemp = value / 256
    value = value - (valtemp * 256)
    // write chr value
    rc = StrSetChr(stWrite$,valtemp,0)
    rc = StrSetChr(stWrite$,value,1)
    //-----
    // SPI
    //-----
    rc=SpiOpen(0,1000000,0,handle)
    // CS (pin select) at low
    GpioWrite(2,0)
    stRead$=""
    rc = SpiReadWrite(stWrite$, stRead$)
    stRead$=""
    rc = SpiReadWrite(stWrite$, stRead$)
    // CS (pin select) at high
    GpioWrite(2,1)
    // close the SPI port twice
    SpiClose(handle)
    SpiClose(handle)
    //-----
    // convert SPI value to Voltage
    //-----
    // read chr value
    valtemp = StrGetChr(stRead$,0)
    value = StrGetChr(stRead$,1)
    value = (valtemp * 256) + value
    value = value * 2500
    value = value / 65535
    // convert value to string
    SPRINT #tx$,value
    tx$ = "\n" + tx$ + "mV"
    // send to Bluetooth
    n = BleVspWrite(tx$)
    text$ = ""
  ENDIF
ENDFUNC 1
```


MEASSY

An instrument for audiophile speaker builders



By **Sergej Koschuch** (Germany)

At some point, even the most passionate loudspeaker builder has enough of the mess and unreliability of improvised instrumentation setups. A cable here, an adapter there, and where did that wire go to? This frustration drove the author to develop a proper instrument. Introducing MEASSY: the measuring instrument for loudspeakers.

For many years the author has been using the ARTA software package [1], which was developed by Ivo Mateljan at the University of Split. To measure the frequency response and impedance of loudspeakers, the author uses a sound card as a signal generator with software-controlled signals and a microphone for signal input. Although the matching ARTA Measuring Box [2] considerably reduces the cable clutter for making measurements, it's only a partial solution. A better approach is to fit all the circuitry and other components needed for a proper measurement setup, including a power amplifier, into an enclosure. That makes a major dent in the number of cables required and frees up space on the bench. MEASSY is an acronym of Measurement Assistant and consists of the following components:

- A balanced, adjustable-gain microphone preamplifier (MPA) with a peak indicator

- A phantom 48-V power supply for condenser microphones or integrated preamps
 - A 30 W power amplifier with clipping indicator
 - A switchable shunt resistor corresponding to the ARTA Measuring Box
 - A USB sound card (optional)
- The above list is essentially the requirements spec for the project. Now let's see what it looks like in practice.

Quality requirements

Before you start drawing circuit diagrams, it's a good idea to consider what you actually want and need and to define some quality requirements. You need separate quality requirements for the MPA and the power amplifier.

Microphone preamp requirements

Aside from the microphone, the MPA is the key component for the dynamic range

and linearity of the MEASSY. The signal to noise ratio (SNR) is largely determined by the input noise level of the MPA. If you want to have a true 16-bit dynamic range (96 dB) with the line input (signal level -10 dBV) of a sound card at a gain of 26 dB, the maximum allowable noise level at the MPA input is 250 nV. When you consider that the thermal noise level of a $100\text{-}\Omega$ resistor is 182 nV, this looks like a difficult requirement. Fortunately, measuring microphones do not have anywhere near this sort of SNR. Even with high-quality devices, the SNR is only about 74 dB referenced to a sound pressure 1 Pa, which corresponds to an acoustic sound pressure level of 94 dB. This yields a calculated equivalent quiescent noise level of 20 dB. With measurements at a sound level of 106 dB, the microphone therefore achieves an actual SNR of 86 dB. In other words, you can knock 10 dB off the 96 dB requirement.

Nevertheless, the corresponding input noise level of $0.8 \mu\text{V}$ (or less) is still a challenge.

Linearity is not much easier. Although there are ICs available with -100 dB THD at unity gain, things look different at a gain of 40 dB . With regard to the necessary gain, the sensitivity of suitable condenser microphones lies in the range of 2 to 20 mV/Pa . Here we use the worst case to calculate the gain: loudspeaker sensitivity of $80 \text{ dB/W}\times\text{m}$ (fairly low), amplifier output power 3 W , measuring distance 3 m , and microphone sensitivity 2 mV/Pa . In this case the sound level at the measuring microphone is 75.2 dB . The gain necessary for a signal level of -10 dBV (316 mV) is therefore 62.8 dB . This means that a gain of 60 dB is sufficient for unfavorable conditions. In practice, assuming loudspeakers with high efficiency, relatively high power and a smaller measuring distance, the gain can even be as low as 0 dB .

Power amplifier requirements

An output power of 30 W is fully sufficient

for measuring the transfer characteristics of home audio systems. The situation is different with sound systems for rock concerts, but that is not what the MEASSY is intended for.

With regard to the SNR, the considerations are similar to those for the MPA, but low power levels also have to be taken into account, so you should assume a requirement of 100 dB . This results in an output noise level of $110 \mu\text{V}$ at the nominal power level ($11 \text{ V}_{\text{RMS}}$ with a 4Ω load). The equivalent input noise level can be found by subtracting the gain — for example, $3.44 \mu\text{V}$ with a gain of 30 dB (amplification factor 32). This noise level is not a significant problem.

Distortion is also not an issue, since even simple circuits can achieve a THD of 0.01% (80 dB below the signal level). Robustness is a more difficult issue. The power amplifier should be short-circuit proof, and due to the small dimensions of the MEASSY, you also have to think about cooling and overtemperature shutdown.

MEASSY electronics

The individual circuits were developed with the above requirements in mind. The power supply is doubtless the least demanding of the lot, so we can start with a description of its special features.

Power supply

The power supply has to provide several voltages: balanced $\pm 15 \text{ V}$ for the MPA, a 48 V phantom feed for the microphone, and balanced $\pm 24 \text{ V}$ for the power amplifier. The supply voltages for the MPA and the microphone should be well regulated, but an unregulated supply voltage with higher load capacity is sufficient for the power amplifier.

As can be seen from **Figure 1**, all you need is a conventional circuit consisting of rectifiers and filter capacitors along with several voltage regulators. Due to the required voltages, you need a power transformer with two 18 V secondary windings. With regard to the power rating, the key factor is the efficiency of a class AB amplifier. For 30 W into a 4Ω or 8Ω load, a toroidal transformer rated

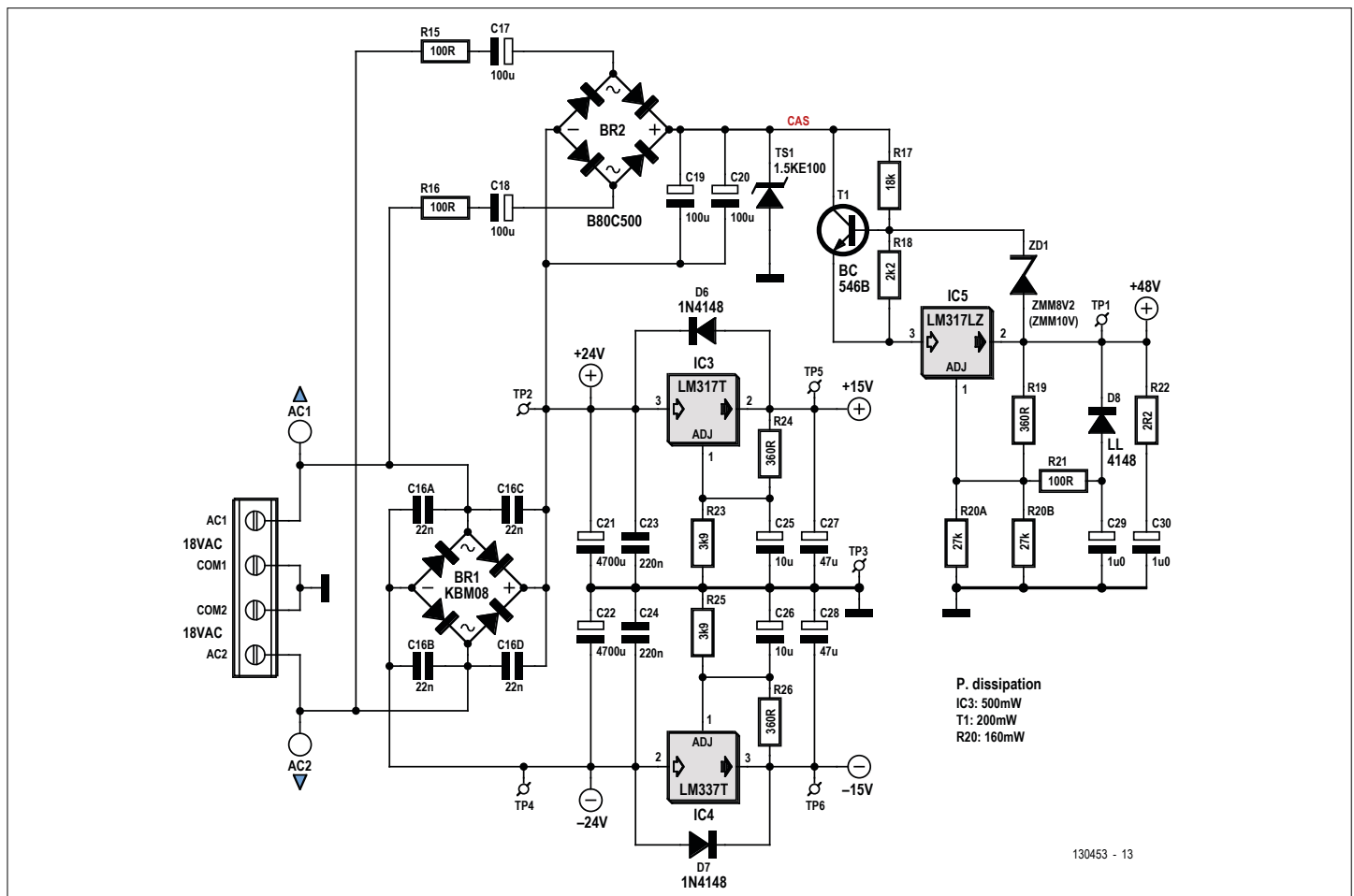


Figure 1. The power supply circuit, which provides unregulated $\pm 24 \text{ V}$ and regulated $\pm 15 \text{ V}$ and $+48 \text{ V}$.

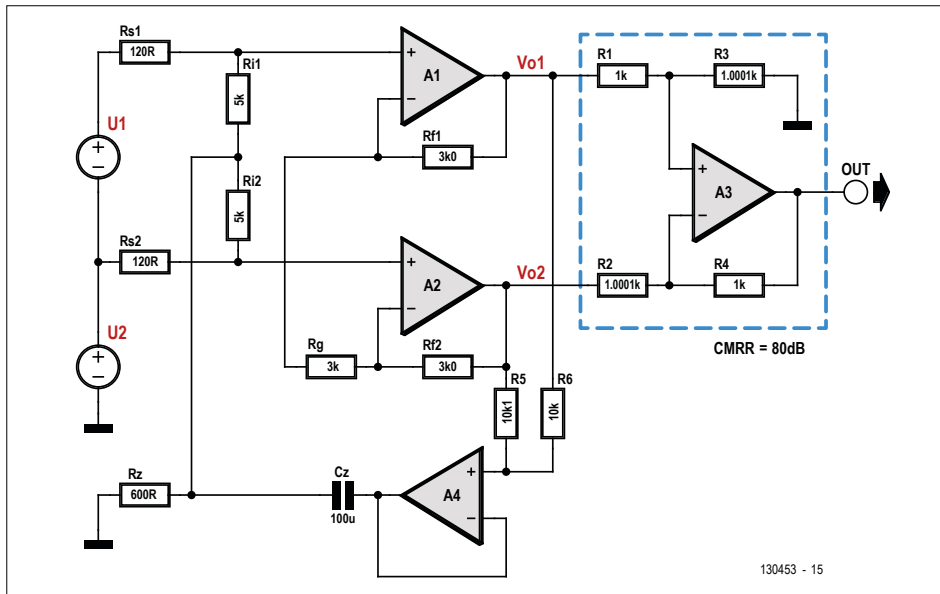


Figure 2. Basic circuit of a balanced microphone preamplifier with enhanced CMRR.

at 50 VA should be more than adequate. The ± 24 V supply voltages for the power amplifier are taken directly from filter capacitors C21 and C22. The ± 15 V supply voltages are provided by the “tried and true” complementary voltage regulators LM317 (IC3) and LM337 (IC4). Capacitors C25 and C26 suppress AC hum and reduce the noise level. The customary protection diodes are not necessary because the two ICs in this circuit do not allow reverse current flow. However, diodes D6 and D7 are provided to protect them against any voltage spikes that find their way onto the supply rails.

For the phantom power supply, we use some clever circuitry instead of a separate transformer. Bridge rectifier BR2 and capacitors C17–C20 form a voltage multiplier (charge pump) that generates 70 V on the positive terminals of C19/C20, nearly three times the peak voltage of the transformer secondary. The no-load voltage can be as high as 80 V. Resistors R15 and R16 limit the peak current. To reduce the required working voltage of C19 and C20 (and thereby their dimensions), their negative terminals are referenced to C21. The downstream voltage regulator is based on a standard application circuit for the LM317 for operation with relatively high voltages [3], since it is not designed to handle 80 V. IC5 is configured as a floating regulator, with an accuracy largely dependent on the current through the regulator (input to output). Resistors R17–R20 are dimen-

sioned to ensure the required minimum current of 2.5 mA under no-load conditions. However, this value is only specified by National Semiconductor (now Texas Instruments). If the current through the regulator is too low, the output voltage will rise. Here it should be noted that the current through Zener diode ZD1 bypasses the regulator (IC5). The value of R17 can be relatively high thanks to the low required current and the gain of T1. As a result, the power dissipation of R20 remains within reasonable bounds. A pair of 0805-SMD resistors wired in parallel can easily handle the 160 mW dissipation.

When loaded with a connected microphone, the current through the regulator is always sufficient. At higher current levels the voltage multiplier would be hard pressed to do its job and the output ripple voltage would rise to more than 10 mV. For this reason the maximum current is limited to 8 mA, which is more than enough for normal measuring microphones. At 8 mA, transistor T1 dissipates 250 mW from its TO-92 package. As an alternative, you could use a BD139 (TO-126 package) for T1.

Microphone preamplifier

The most logical place to test loudspeakers is in their normal listening position. The necessary cable length and associated noise problems make unbalanced microphone connections unsuitable. Besides, most measuring microphones

have a balanced signal output, such as the commonly used model MM1 from Beyer Dynamic or the TR-40 from Audix, both priced at about €180.

The specified requirements can be fulfilled nicely by the type INA163 instrumentation amplifier from Burr Brown (now part of TI). As a special feature, it has additional driver stage outputs that can be used to boost the common mode rejection ratio (CMRR). Due to the need for a phantom power supply, the input is not only low-impedance but also connected to ground, which is unfortunate. The CMRR is therefore determined by the imbalance in the apparent output impedance of the signal source, regardless of the balance of the signal input. Unlike the situation with an ungrounded transformer output, with direct galvanic coupling an imbalance of more than 20 Ω can occur due to tolerances and long cables, which results in a relatively poor CMRR of about 40 dB with typical impedances.

A simple countermeasure is to use bootstrapping to raise the input impedance. **Figure 2** shows the basic circuit of this sort of balanced amplifier. Opamp A4 feeds the sum signal from outputs Vo1 and Vo2 back to the input. Capacitor Cz prevents input saturation with this form of positive feedback. Summing with resistors R5 and R6 is intended to strongly suppress the difference signal in order to avoid bootstrapping the difference signal. As a result of the positive feedback, the signal currents through Ri1 and Ri2 are very small and the common-mode input impedance is much greater than the physical resistor values, but the differential input impedance ($R_{i1} + R_{i2}$) remains the same.

It should be noted that due to the electrometer configuration of A1 and A2, the common-mode gain of the first stage is fixed at 1, regardless of the value of resistance Rg. Now the 80 dB CMRR of the second stage can be maintained over a wide frequency range. The lower cutoff frequency is determined by the high-pass network Cz/Rz, while the upper cutoff frequency is determined by the bandwidth of A4. With this arrangement, the quality of the cable is practically insignificant. The precise mode of operation of this circuit topology is described in detail in an AES article [4].

The basic circuit of Figure 2 is incorporated in the actual MPA circuit shown in **Figure 3**. There the bootstrapping

is extended to include two other paths for HF filtering and the phantom supply. Resistors R3 and R4 inject the 48 V supply voltage; they should be selected for matched values. The combination of C8, R12 and the narrow bandwidth of IC2A forms an HF filter. Additional filtering of the phantom supply voltage is provided by C11. A Spice simulation of the circuit with an imbalance of $20\ \Omega$ and a gain of 20 dB yielded a consistent CMRR of more than 60 dB from 70 Hz to 80 kHz. IC2B is wired as a DC servo to eliminate any DC offset at the output. The cutoff frequency of the integrator is 2 Hz. IC2 should have FET inputs. A detailed explanation of how the protection diodes work would take too much space here. However, you can read more about it at [5].

For a linear amplifier gain control in dB units you need a potentiometer with a reverse logarithmic characteristic, which is very hard to find. With an adjustment range of more than 40 dB, it's not possible to achieve this by "tweaking" a linear potentiometer. However, continuous adjustment is by no means necessary because modern USB sound cards have integrated programmable gain amplifiers (PGAs), which can boost the signal by at least 10 dB without noticeably increasing the noise level. A gain control with discrete settings in steps of 6 to 10 dB is therefore sufficient. A two-gang selector switch with six positions, with the two gangs wired in parallel for reliability, allows the gain to be set over the range of 6 to 48 dB in six 8-dB steps.

Peak indicator

The peak signal level indicator is implemented with an LM3915 and a full-wave rectifier (see **Figure 4**). This part of the circuit is conveniently fitted on a small, separate PCB along with the rotary switch for the gain setting, which is mounted vertically behind the front panel. IC2 operates in dot mode to minimize noise on the supply rail and reduce power consumption. Because IC2 is connected to the MPA supply rail and the individual LEDs have a forward voltage of approximately 2.2 V, the maximum number of LEDs that can be used here is six. To increase the display range to 6 dB for small signals, two IC outputs are combined for each of the first four LEDs (LED1 to LED4). The current through resistors

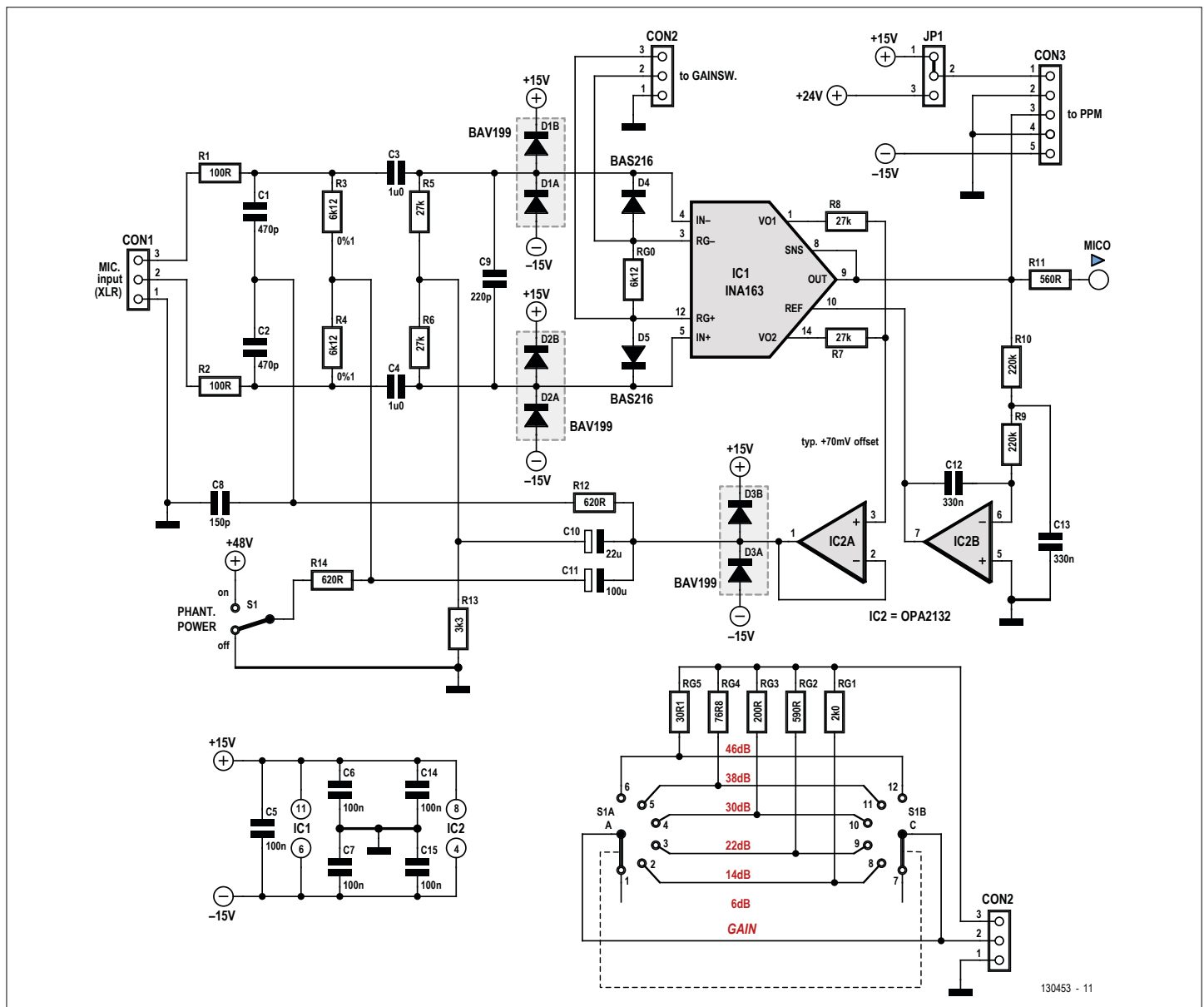


Figure 3. Microphone preamplifier circuit based on an instrumentation amplifier IC.

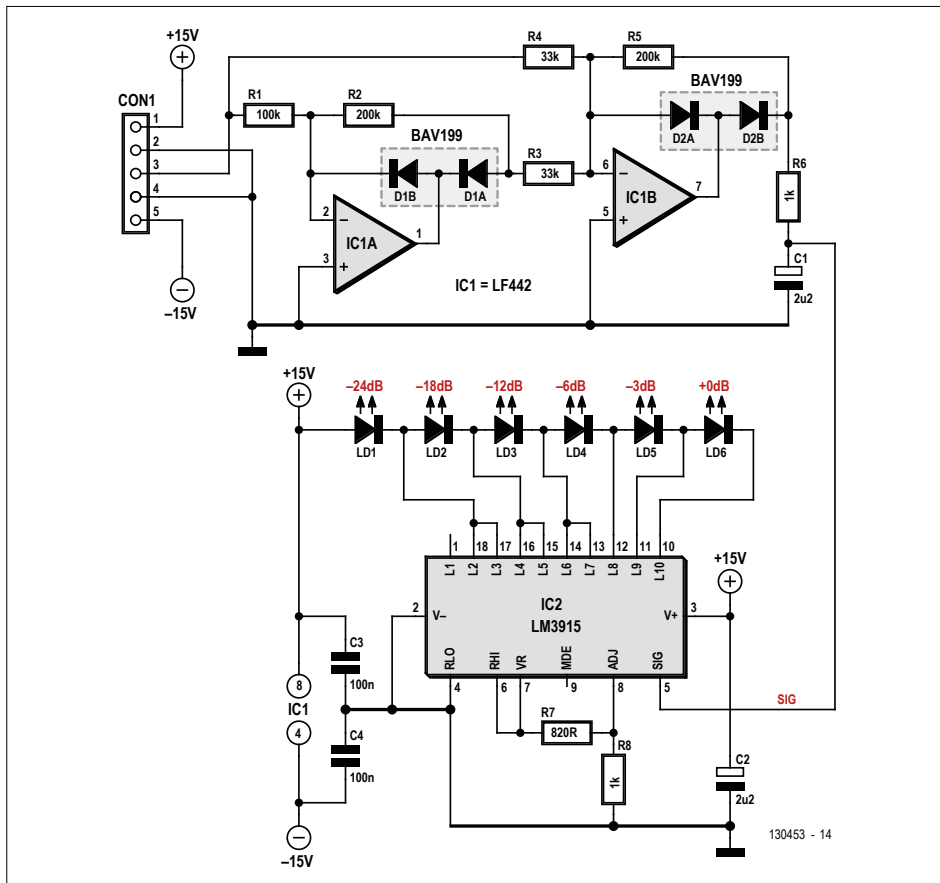


Figure 4. The peak indicator consists of an LED bar driver and a full-wave rectifier.

R7 and R8 (16 mA) determines the brightness of the LEDs as well as the total signal level range. The maximum level for SIG is set to 2.72 V. Together with the gain of the rectifier (IC1), the output signal level of the MPA required for maximum signal amplitude is 447 mV_p (-10 dBV), which is the normal line level for consumer electronics. This gives the MPA a healthy headroom of 20 dB.

Because the output of the MPA has no DC offset, the conventional rectifier circuit built around the two opamps in IC1 can be DC coupled. The amplification factor of the two stages is 6.06. Due to the different impedance values of the two stages, the diode leakage currents cause a small gain error (less than 0.5 dB) between the two half-waves, which is negligible in practice. The rise time 2.2 ms, which is determined by the time constant of R6/C1. The fall time is 0.44 s (determined by (R5 + R6) x C1). Fall times shorter than 0.4 s are generally regarded as undesirable.

Power amplifier

Integrated power amplifiers are a good

choice for output power levels below 50 W. Modern IC power amplifiers have excellent specifications, and they are compact and inexpensive. Two options here are the LM3886 from TI and the TDA7293 from STM. The former is short-circuit proof, but the STM IC has better specs and its Clip and Short Detect (CSC) output, which can be used to drive a clipping indicator, is a major advantage because it makes distortion measurements more reliable. The author also knows from experience that the TDA7293 can survive output shorts if the supply voltage and signal level are low. Operational reliability is enhanced by a supplementary short-circuit detector.

Figure 5 shows a standard application circuit for the TDA7293. The gain is set to 30 dB by resistors R29 and R30. No capacitor should be connected in parallel to R29. A Boucherot network (R33/C40) at the output improves stability. Capacitor C40 is a multilayer type instead of a film type, due to the better specs of the former. Resistor R33 should be a low-inductance type. The combination of capacitors C33a and C33b (with bias voltage

provided by R31) forms a discrete bipolar electrolytic capacitor with very good characteristics. Its capacitance puts the lower cutoff frequency at about 11 Hz. Capacitors C33a and C33b should have very low leakage current, as otherwise the output offset voltage may rise slightly due to the current through R31. The R28/C32 network at the input forms a low-pass filter that attenuates signals above 80 kHz.

The CD output on pin 5 is an open-drain type. It provides a precise indication of amplifier saturation or clipping. This output drives a non-retriggerable monostable built around IC8b and is active at distortion levels as low as 0.5%. Even extremely short clipping events therefore cause the red part of the bicolor LED to blink for 0.5 s, driven by T6. In combination with the green "Operating" indication, LED1 is thus lit yellow in the event of clipping. The CLIP signal is also connected to IC8a at the junction of R51/C46. This circuit detects the duty cycle. At 40% (zero reference level) or more, flip-flop IC8a is constantly set and the mute pin of IC6 is pulled to ground by T4. The red part of the bicolor LED is then lit by T5, but the base current for T7 is cut off and the green part is dark. The resulting red indication signals a probable short-circuit condition.

The response time is determined by R51/C46 and R49/C44 and depends on the mark/space ratio: 36 ms at 40%, 24 ms at 50%, and 13.5 ms at 75%. Conventional solutions with relays take approximately 15 ms. A duty cycle less than 36% is interpreted as saturation (full clipping) and corresponds to a THD of approximately 9%. If a short-circuit condition is detected, the amplifier must be switched off and then switched on again. The external reset signal for IC8a comes from the collector of T2, so short-circuit detection is enabled shortly before the amplifier output is unmuted. This prevents false detection during power-up.

The +6 V auxiliary voltage is generated by voltage regulator IC7. The circuitry around T2 and T3 provides a switch-on delay of approximately 2.5 s and immediate shutdown of the amplifier. The different resistance values (R45 + R46 versus R47 + R48) result in a time delay between the signals on the STBY and MUTE inputs of IC6. Noise suppression utilizes the Standby function. The series resistor for IC7 (R40) has a relatively

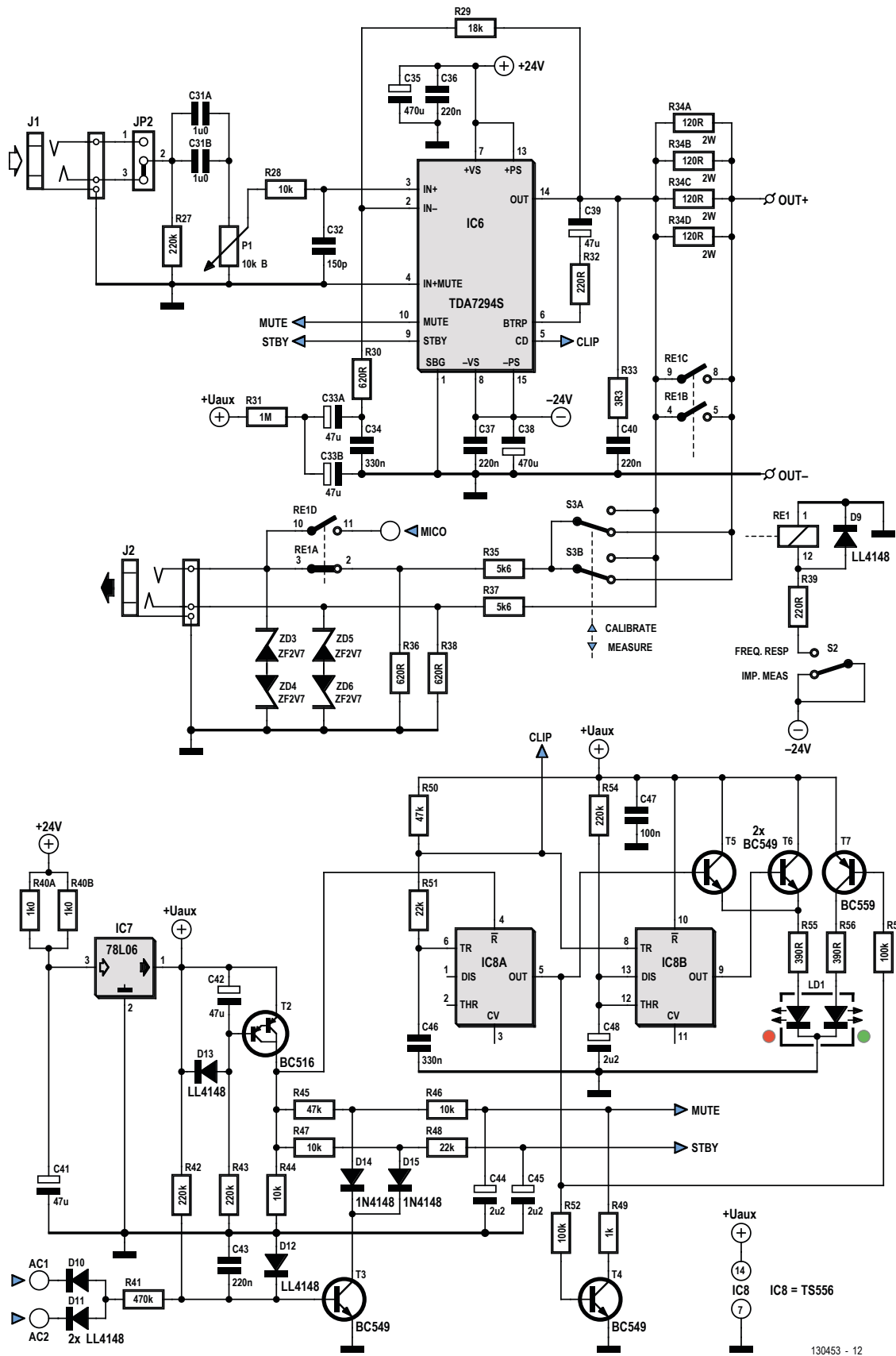


Figure 5. Integrated power amplifier with clipping detector and auxiliary voltage generator.

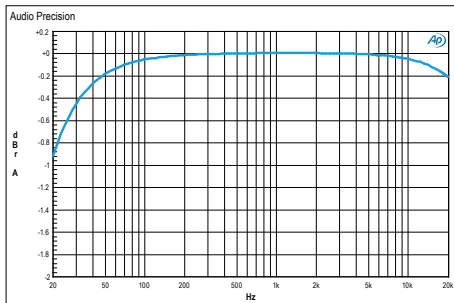


Figure 6. The frequency response of the power amplifier at 1 W output into an 8 Ω load.

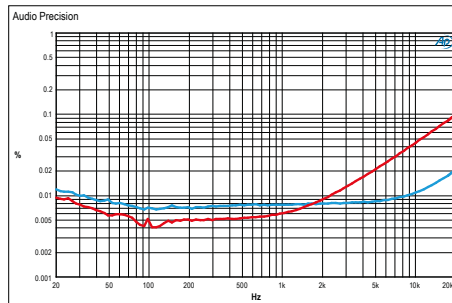


Figure 7. THD + N at 1 W (blue) and 20 W (red) into 8 Ω.

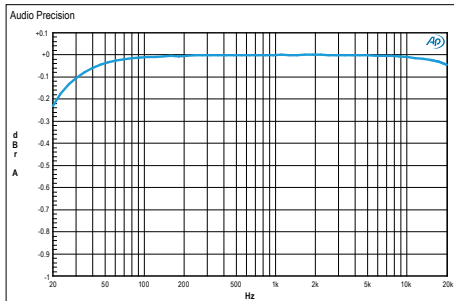


Figure 8. The frequency response of the preamplifier with the same scale as in Figure 6.

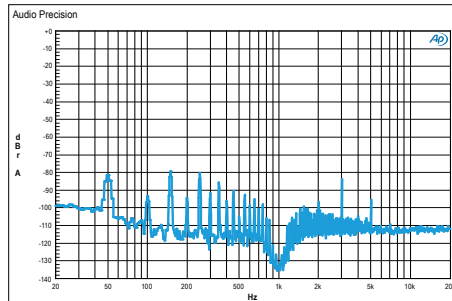


Figure 9. The frequency spectrum of the MPA with a 1 kHz test signal (suppressed).

large value to limit the power dissipation of the transistor to less than 250 mW. The LED1 currents are also limited to 10 mA per diode for the same reason. The circuitry around S2, S3 and Re1 incorporates the measuring functions of the ARTA Measuring Box, with R34 as the measuring resistor. It consists of four metal-oxide resistors, each rated at 2 W. That is sufficient for the power dissipated

by the effective 30 Ω resistance. Unlike the original Measuring Box, S2 is complemented by a high-quality relay, which is not readily available. Voltage dividers R35/R36 and R37/R38 are suitable for an amplifier output power of 30 W (-20 dB at $12 V_{RMS}$).

A few words about cooling: The 50 VA toroidal transformer limits the maximum output power to less than 35 W, regard-

less of the load impedance. However, loudspeaker measurements are rarely made at full output, but instead about 3 dB lower, corresponding to an output power of approximately 20 W. The efficiency of a class AB amplifier is about 43% at moderate load levels, so the dissipation is around 12 W. With a maximum ambient temperature of 45 °C and a thermal shutdown threshold of 125 °C, the allowable temperature range is approximately 80 °C. The permissible thermal resistance is therefore approximately 6.5 K/W.

A properly mounted TCA7293 IC has a thermal resistance (die to heat sink) of 2 K/W. This leaves a surprisingly high value of 4.5 K/W for the heat sink to air thermal resistance. Measurements do not require continuous operation, but a fan might be a good idea if you expect to make prolonged measurements. An example of a suitable heat sink is the Fischer SK574, with a length of 75 mm.

Practical aspects

The MEASSY is designed for an input signal level of $316 mV_{RMS}$ (-10 dBV). That is suitable for many internal sound cards, but not for the Transit USB model from M-Audio used by the author. The USB audio codecs in the Transit USB work with a 3.3 V supply voltage. The inputs and outputs are unbalanced, and the signal levels are in the range of 0.72 to $1.35 V_{RMS}$, which is around 0 dBV. With this sort of sound card the MEASSY “throws away” about 7 dB of dynamic range, even though the MPA output can deliver much higher levels. If you want to utilize the maximum dynamic range in this situation, you can change R8 to 1.8 kΩ on the separate PCB. Then the -3 dB LED will light up at $700 mV_p$ to indicate full amplitude at $700 mV_{RMS}$. The same applies to R35/R36 and R37/R38. At maximum power the signal level at the amplifier output is $12 V_{RMS}$. For impedance measurements, this can be achieved with the output signal from a sound card with an input signal level of 0 dBV, attenuated by 20 dB.

The MEASSY was measured in the Elektor Labs, independently of the author. **Figure 6** shows the frequency response of the power amplifier at 1 W output into an 8 Ω load, which fits very well with the intended use. A frequency response from 1 Hz to 100 kHz is not necessary because measurements are not made in that range. Note the very high vertical resolution of

The Author

Sergej Koschuch has an engineering degree and more than 20 years' experience as an analog designer for integrated circuitry at STMicroelectronics, Micronas and Infineon. He has also maintained an active interest in discrete circuit design, which is still common and useful in loudspeaker construction.

Web links

- [1] ARTA software: www.artalabs.hr
- [2] ARTA Measuring Box www.artalabs.hr/AppNotes/AP1_MessBox-Rev3Eng.pdf
- [3] LM317 application note: www.ti.com/lit/an/snva583/snva583.pdf
- [4] Circuit theory: www.thatcorp.com/datashts/AES6261_New_Balanced_Input_IC.pdf
- [5] Protection measures: www.thatcorp.com/datashts/AES7909_48V_Phantom_Menace_Returns.pdf
- [6] PCB files: www.elektormagazine.com/130453

the chart (+0.2 dB to -2 dB). Of course, we also measured the distortion (THD + N). The blue curve in **Figure 7** shows the distortion at 1 W and the red curve the distortion at 20 W, both with an 8 Ω load. At 1 W the SNR is 80 dB from 30 Hz to more than 8 kHz. At 20 W the distortion rises at higher frequencies, as expected. Overall, the results are suitable for the intended measurement purposes.

Not surprisingly, the frequency response of the MPA is distinctly better, as can be seen in **Figure 8**. The frequency spectrum at the MPA output with maximum gain and a 1 kHz, 4 mV test signal (suppressed) is shown in **Figure 9**. The first thing you notice is the AC hum and its harmonics. The harmonics are at around -80 dB, and the non-harmonics are much lower. The third harmonic of the test signal is visible at -84 dB, but the other harmonics are negligible. The THD + N is 0.03% with a bandwidth of 22 kHz. At low gain (amplification factor 30) with a relatively high signal level (12.65 mV), the distortion level rises correspondingly to 0.013%.

Summary

The author built a very attractive MEASSY prototype, as can be seen in **Figures 10** to **12**. Figure 10 shows the MEASSY in an AUS12 case from Teko, with a very tidy layout for the aluminum front panel. For new construction, the author would use Cinch (RCA) sockets instead of the two 3.5-mm jack sockets, for quality reasons. The most striking feature on the rear panel (Figure 11) is the gold-plated loudspeaker terminals. A standard three-pin XLR socket handles the microphone connection.

The PCB layout files for this project are available on the web page for this article [6]. The author recommends that only audiophiles with sufficient experience, who also understand the content of articles [1] and [2], attempt to construct the MEASSY.

If you would like to see the MEASSY implemented as an Elektor project, please indicate this by sending an e-mail to Jan on: editor@elektor.com. In that case PCBs will be designed in the Elektor Labs and a DIY project will be developed, including a detailed description of how to use the MEASSY in practice to measure the impedance, frequency response and other characteristics of loudspeakers. ◀

(130453-1)



Figure 10. The fully assembled MEASSY, ready to use.

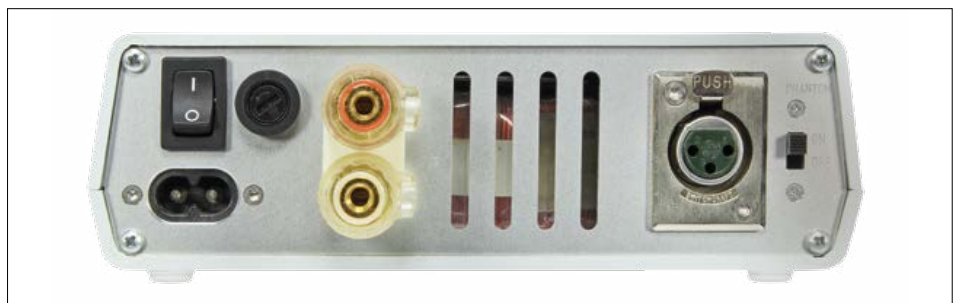


Figure 11. MEASSY rear panel with connectors for AC power, loudspeaker and microphone.

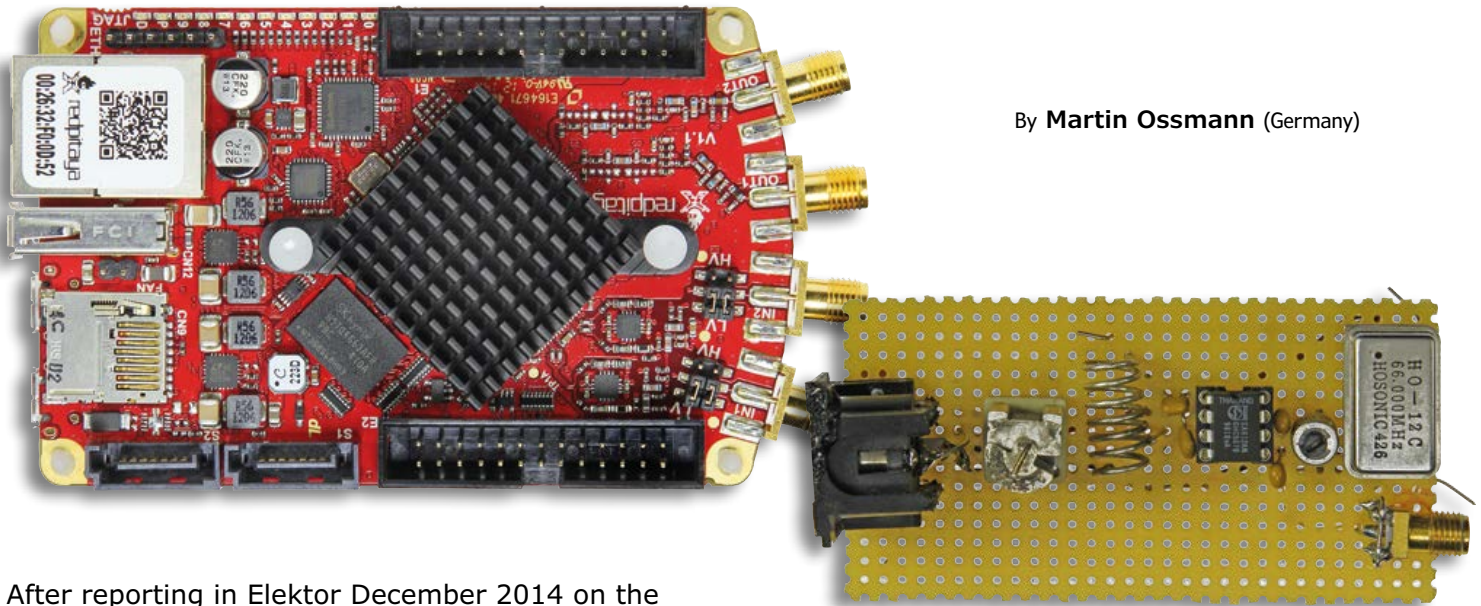


Figure 12. A good view of the interior of the MEASSY. Note that this is an early prototype and differs slightly from the final version.

Enhanced FM Stereo on Red Pitaya

Advanced instrumentation platform 'on the air'

By Martin Ossmann (Germany)



After reporting in *Elektor* December 2014 on the versatile 'Red Pitaya' board we now discuss using the instrumentation platform to produce a stereo VHF receiver. Paramount in this case is not the practical application — you can buy an FM radio cheaply after all — but rather the principles applied. A normal broadcast FM signal (stereo plus R(B)DS) with a bandwidth of 400 kHz [1] calls for digital processing of corresponding efficiency and capability. Red Pitaya measures up to this challenge admirably.

To call Red Pitaya nothing more than a USB oscilloscope would be a shallow understatement. In the introductory article in *Elektor* [2] we explained how this programmable open-source instrumentation platform could also be deployed as a signal generator, spectrum analyzer, oscilloscope, PID controller and much more besides. Numerous programs for doing this are available free. Enhancing the high-performance system-on-chip, with a Xilinx ZC7Z010 (dual-core ARM9 CPU), fast FPGA and a plentitude of RAM — running under Linux (on an SD card like the Raspberry Pi) — we also have high-speed A/D and D/A converters to provide an interface to the analog world. Red Pitaya also provides, as appropriate for a system of this kind, numerous other interfaces — from universal I/Os, through RS-232 and USB, all the way up to Ethernet.

The concept

Red Pitaya can digitize signals up to around 50 MHz. For this reason a simple front-end (as seen in **Figure 1**) is required to convert the broadcast band between 88 and 108 MHz down to the region between 22 and 42 MHz. An NE612 is deployed as mixer, using the 66 MHz crystal local oscillator LO1.

The signal from the ADC in Red Pitaya is sampled at 125 MS/s and fed to the FPGA. It's in the FPGA that the first stage of our FM receiver is achieved, mixing down to baseband (zero IF) and the filtering. This produces a signal with a sample rate of around 244 kHz at the input to the CPU, where the actual demodulation takes place. The stereo decoder is transformed as C code, leaving just enough processing power in hand for an RDS demodulator/decoder.

Inside the FPGA

The mixing process in the FPGA is carried out using a DDS oscillator, which follows the signal generator provided with Red Pitaya. Instead of passing the output to a D/A converter, this signal is used for mixing within the FPGA. Sine and cosine signals are obtained by read-out from the table, using an offset that corresponds to 90 degrees. This produces a so-called I/Q mixer [3].

The two components calculated in this way — I (In-phase signal) and Q (Quadrature phase signal) — enable us to determine the amplitude and phase of the signal being received.

If A is the amplitude of the signal and p its phase, then the following is valid:

$$I = A \cos(p)$$

$$Q = A \sin(p)$$

After mixing, the IQ signals are low-pass filtered and divided down from the high sampling rate of 125 MS/s to a rate of around 244 kS/s (by the factor $R=512$). For filtering and dividing we deploy CIC filters, as described in [4] and [5] (**Figure 2**). CIC (cascaded integrator-comb) filters create so-called 'moving averages', in which the filters are divided into two sections, between which the rate reduction ensues.

Since a CIC filter of this kind uses only additions, subtractions and time delays for clock-timing (represented by z^{-1}), it can be programmed using an FPGA in a simple and non resource-hungry manner. The critical lines in Verilog are:

```
reg [width-1:0] CICint1 ;
reg [width-1:0] CICint2 ;
reg [width-1:0] CICstore1 ;
reg [width-1:0] CICdiff1 ;
reg [width-1:0] CICstore2 ;
reg [width-1:0] CICdiff2 ;

always @(posedge cic_clk_i) begin
if ( inp_strobe_i ) begin
CICint1 <= CICint1 + inp_data_i ;
CICint2 <= CICint2 + CICint1 ;
end end

always @(posedge cic_clk_i) begin
if ( out_strobe_i ) begin
CICdiff1 <= CICint2-CICstore1 ;
CICstore1 <= CICint2 ;
CICdiff2 <= CICdiff1-CICstore2 ;
CICstore2 <= CICdiff1 ;
end
end
assign out_data_o = CICdiff2 ;
```

Two filters of this kind are stacked one following the other, with a sample-rate reduction of 32 and 16. In this way we achieve a sampling rate of $125 \text{ MS/s}/(32 \times 16) \approx 244 \text{ kS/s}$.

The quadrature signal filtered in this way is now fed into a FIFO register, to be read out into a C program for further processing. Incidentally, the IQ signal with a sampling rate of 244 kS/s can accommodate an FM signal of 244 kHz bandwidth as it consists of two signals, both sampled at 244 kS/s. According to sampling theory, a single signal could of course convey a bandwidth of only 224 kHz/2.

To check the function of the front-end, it is frequency-swept with a signal generator. The amplitude of the signal in the FIFO register is determined computationally by the ARM CPU and output logarithmically via one of the DACs. The result can be seen in the sweep curve of the filter in **Figure 3**.

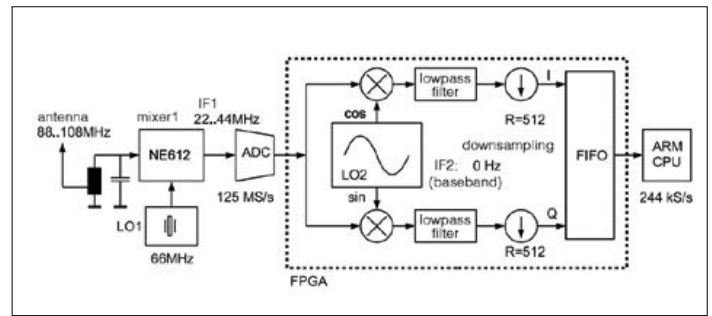


Figure 1. Double mixing using hard and software.

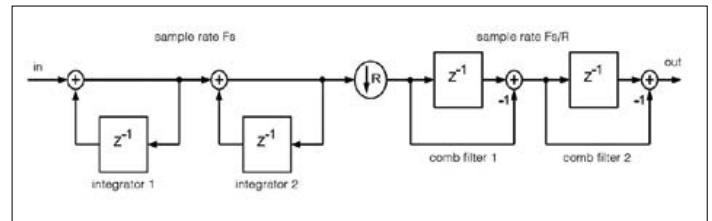


Figure 2. Block diagram of the CIC filter.

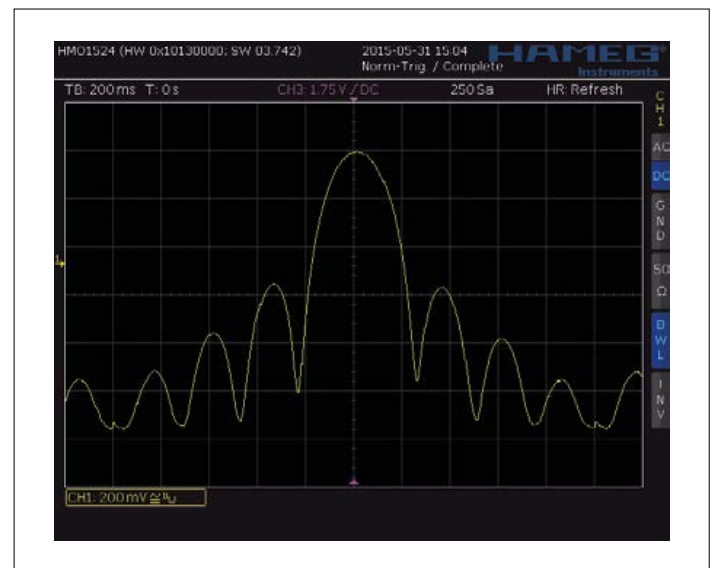


Figure 3. Passband characteristics of the FPGA front end.

The horizontal scale in the illustration amounts to 200 kHz/div and the vertical to 10 dB/div. You can already recognize the passband characteristic of the front-end. This is an example of how we can achieve a good insight into the FPGA code with the help of C software and D/A converters.

FM demodulation

As the pair of IQ signals already contains information on the instantaneous phase of the signal, we can, as shown in **Figure 4**, derive the actual phase simply using the C function $\text{atan2}(Q, I)$. The current frequency is calculated by establishing the difference between consecutive values. In the process we naturally need to take care that at the transition from n to $-n$ the phase alters only seemingly. After frequency determination we have a low-pass filter.

In **Figure 5** we can see the instantaneous phase (above, in yellow) and the current frequency (below, in blue) for an FSK (frequency-shift keyed) signal with square-wave frequency keying. After each of these, according to whether the frequency is positive or negative, a rising or falling saw-tooth is produced as the phase response. **Figure 6** shows the spec-

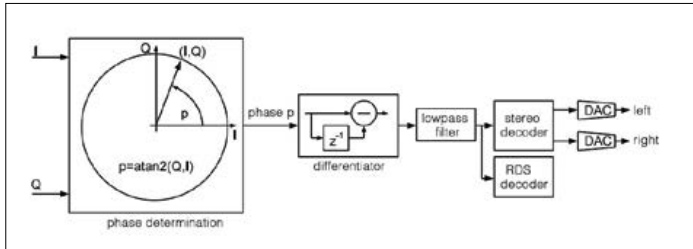


Figure 4. FM demodulation using C language software.

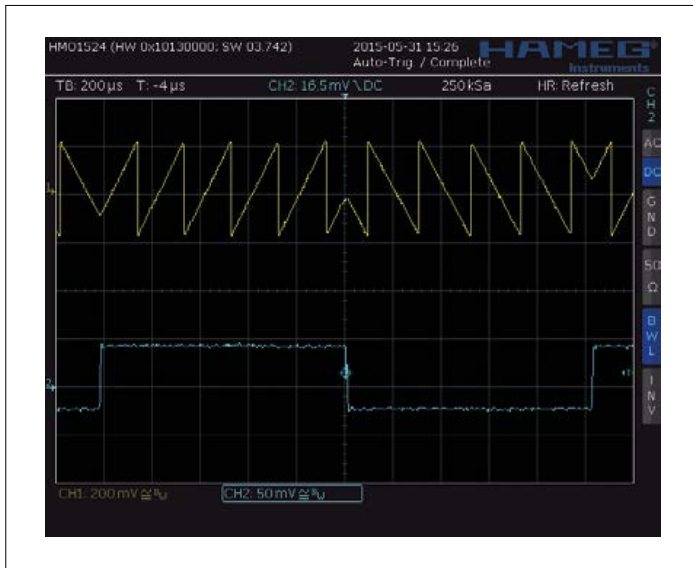


Figure 5. Instantaneous phase (yellow) and frequency (blue) in FSK.



Figure 6. Spectrum of the received MPX signal.

trum of a received MPX signal, evaluated 'on the hoof' by Red Pitaya using FFT and output by both D/A converters into an x-y representation on an oscilloscope. The 19 kHz pilot tone is clearly recognizable as a narrow, tall spike and to the left of this, the spectrum of the L+R audio signal. To the right of this, symmetrically at 38 kHz, we find the L-R signal. The two small humps at far right indicate the RDS signal. With live spectral displays of this kind, the high processing power of Red Pitaya is of course especially noticeable.

Stereo decoder

Let's return to Figure 4... After low-pass filtering the signal reaches a stereo decoder, which is once again programmed in C.

```

phase=atan2(inputI,inputQ) ;// current phase
frequency=phase-lastPhase ; // phase
difference=frequency
lastPhase=phase ;
if ( frequency < -M_PI ) { frequency += twoPi ; } //
2-Pi Periods
if ( frequency > M_PI ) { frequency -= twoPi ; }
MPXsignal = 2000.0 * frequency ; // MPX signal ready

phase19=phase19+frequency19+vco19 ;// 19 kHz Pilot
phase, vco=Steuerspannung
if ( phase19>twoPi ){ phase19 -= twoPi ; } // 2-Pi
periods
integrate19 += cos(phase19)*MPXsignal ; //
multiplying and summing

int LminusRraw=2.0*sin(2*phase19)*MPXsignal ; // L-R-
mixing with 38 kHz
int LplusRraw=MPXsignal ;

cicInSample(..., LplusRraw) ; // L+R filtering
cicInSample(..., LminusRraw) ;// L-R filtering

```

The details of the stereo decoder are shown in **Figure 7**. A PLL reconstructs the 19 kHz auxiliary carrier. Using the 19 kHz oscillator implemented in this way, at the same time we can take care of the 38 kHz auxiliary oscillator, with which the L-R signal is mixed down to baseband. Low-pass filtering produces the L+R signal from the MPX signal. The L-R signal is filtered using the same low-pass. The signals of the two stereo channels arise out of L+R and L-R by means of addition and subtraction. The great thing with Red Pitaya is that you can portray test signals with the D/A converters. **Figure 8** shows (above) the L-R signal in yellow plus (below) the VCO control voltage CTRL in blue, and indeed on the left-hand side with the control circuit disabled (with switch S not closed). The control voltage has a sine-wave form, in which its frequency is the difference frequency between the transmitted and local 19 kHz pilot tone. Simultaneously the L-R signal is amplitude modulated at double the frequency. The amplitude varies according to how well the phase of the local 38 kHz auxiliary carrier agrees with the transmitted carrier.

More or less in the center of the picture the control circuit is switched in. After a brief transient oscillation the PLL locks in and the L-R signal is recaptured with constant amplitude. With the stereo decoder we have now produced a fully functioning

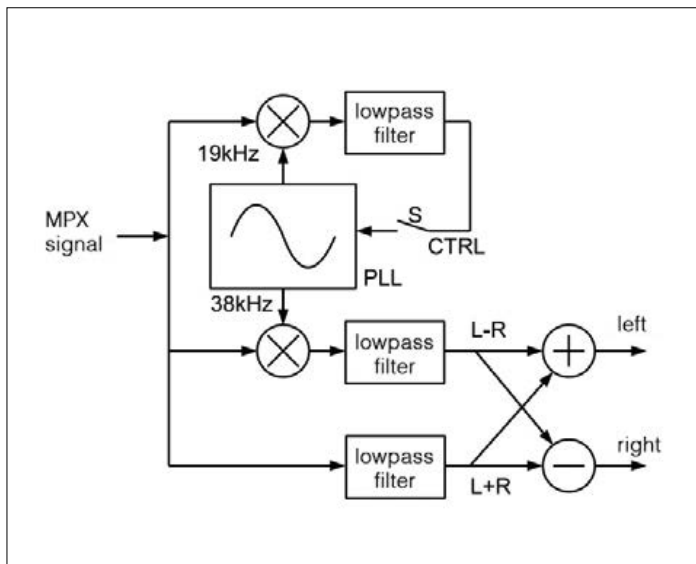


Figure 7. Block diagram of the stereo decoder.



Figure 8. Inner workings of the stereo decoder.

FM receiver in SDR using Red Pitaya. All we are missing now are the final touches (RDS demodulation and decoding).

RDS decoding

Figure 9 shows the block diagram for extracting RDS information. Since the 57 kHz RDS signal contains no spectral components itself, conventional PLL chips cannot be used for recovering the 57 kHz auxiliary carrier. Instead we need to employ a 'Costas Loop' circuit, which in effect extracts the carrier from the sidebands.

As we start implementing RDS decoding, the ARM CPU begins to reach its limits. However, by applying DDS table techniques instead of complex trigonometric functions (sine, cosine) for implementing the various mixer oscillators and switching in speed optimization of the compiler, we can claw back sufficient 'headroom' to enable simultaneous stereo and RDS demodulation with the CPU of Red Pitaya at 244 kS/s. Linked to the demodulation we have sample-rate recovery as well as frame detection and synchronization with error correction. Following this we can finally display the data.

In a way events have now come full circle for the author, whose software RDS decoder [6] saw its first publication in Elektor 25 years ago.

Final thoughts

A project of this kind is obviously not completed in a single weekend. It's important to progress in stages, checking the functionality of each individual block (mixers, filters, rate reduction and control circuits) one at a time. When you do this, it is worth outputting test signals from the FPGA or the C software using the D/A converter and evaluating these with the 'scope. In this respect the process for a software-based project resembles classic hardware development. In the same way you can feed in test signals as and when required (from signal generators) into the software via the A/D converters, for instance to frequency-sweep filters. Or let the software generate test signals itself. ◀

(150326)

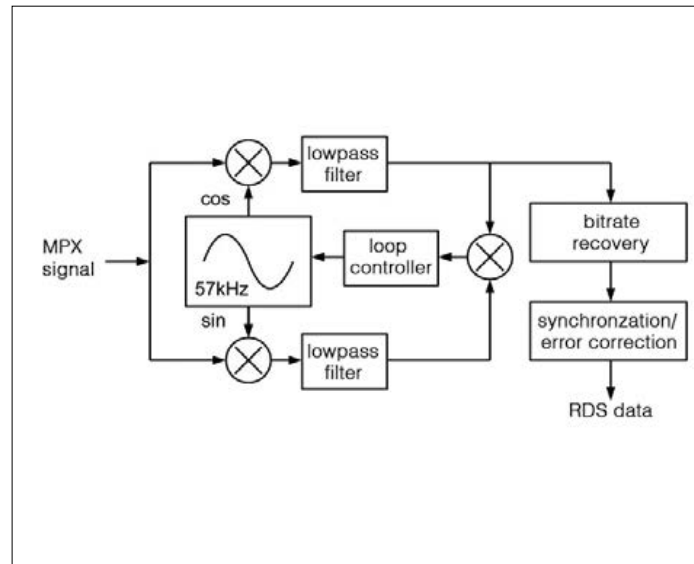


Figure 9. Block diagram of the RDS decoder.

Web Links and Documentation

- [1] https://en.wikipedia.org/wiki/FM_broadcasting
Note: Radio Broadcast Data System (RBDS) is the official name used for the U.S. version of RDS.
- [2] www.elektormagazine.com/140277
- [3] Ellis, Michael, Using Mixers in Radio Communications, <http://michaelgellis.tripod.com/mixerscom.html>
- [4] Donadio, P., CIC Filter Introduction, <http://home.mit.bme.hu/~kollar/papers/cic.pdf>
- [5] Ossmann, Martin: AVR Software defined Radio, E-book, Elektor International Media, www.elektor-magazine.com/fileadmin/E-book_AVR_Software_Defined_Radio.pdf
- [6] Ossmann, Martin: RDS Software Decoder, Elektor December 1989

The ARM board used in our course *From 8 to 32 bits: ARM Microcontrollers for Beginners* [1] is an undisputed bestseller in the Elektor Shop. No wonder, because for around €40, £30 or US\$45 the SAM D20 Xplained Pro offers a powerful platform for developing your own embedded projects [2]. Included is a SAMD20J18 controller with a generous 256 KB of Flash and 32 KB RAM [3], whilst to simplify development a handy Debugger by the name of EDBG is also on board [4]. After downloading the cost-free development environment

Atmel Studio 6 [5], you can get cracking straightaway with program construction, as no additional hardware or software tools are required. For checking out the various controller features like UART, I²C, ADC and similar three expansion connectors are provided in the form of 2x10-pin header strips. In addition to 3.3 V and Ground in each case, 16 Controller pins are provided, whose functions can be preset in the normal way using Controller Registers. All of these pins can behave as digital inputs and outputs, in many cases

also as analog inputs for measuring voltages. Many of these pins have interface functionality in addition. **Figure 1** shows the three expansion connectors with their pin numbering and respective Port pins. The logical functions given are a selection we have made (and among these, those that can be preset). The lower half of each connector contains interface pins for I²C, UART and SPI, above which you'll see the digital inputs and outputs (GPIO), analog inputs (AIN) along with pins provided for timer and counter functions (TC).

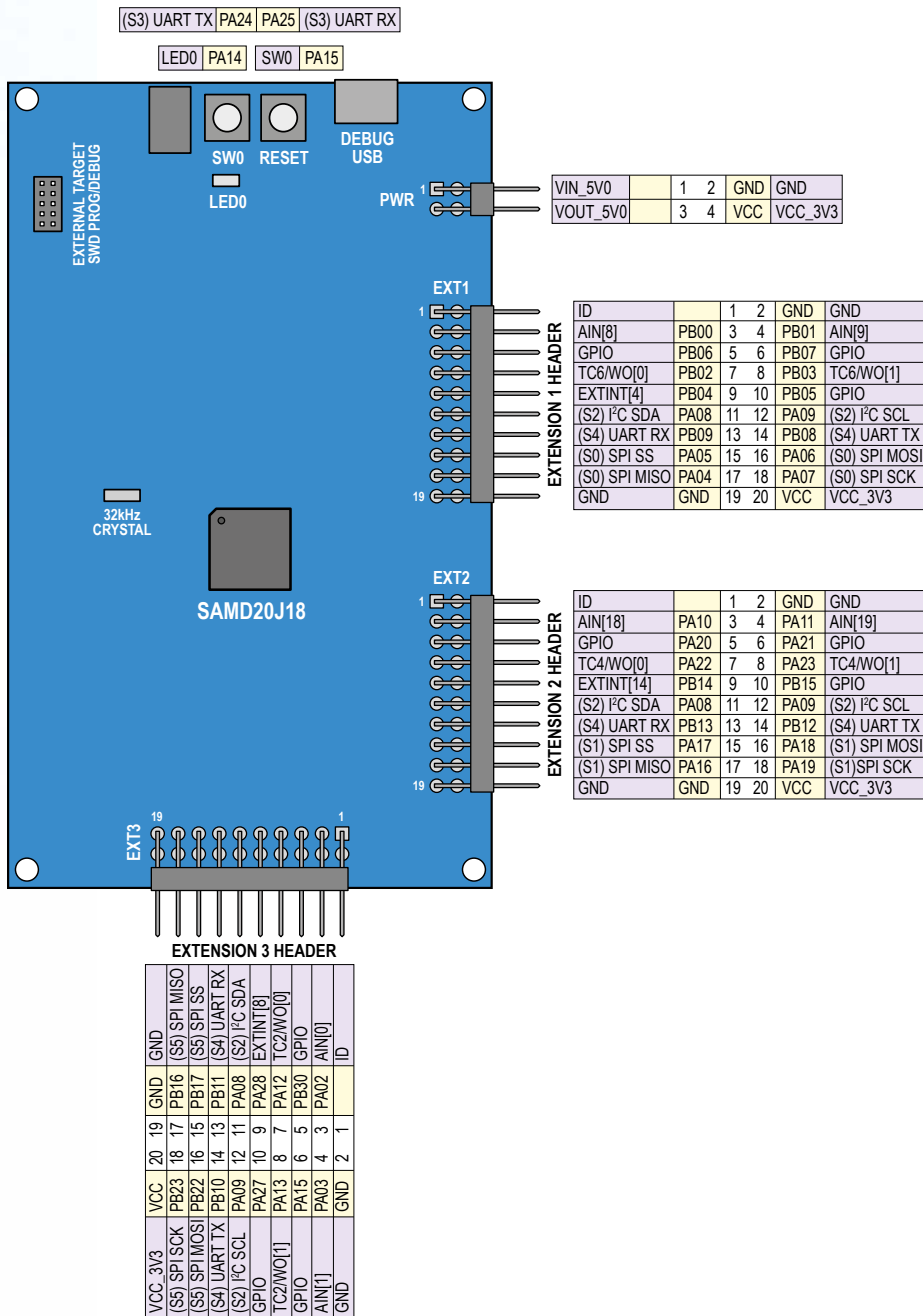
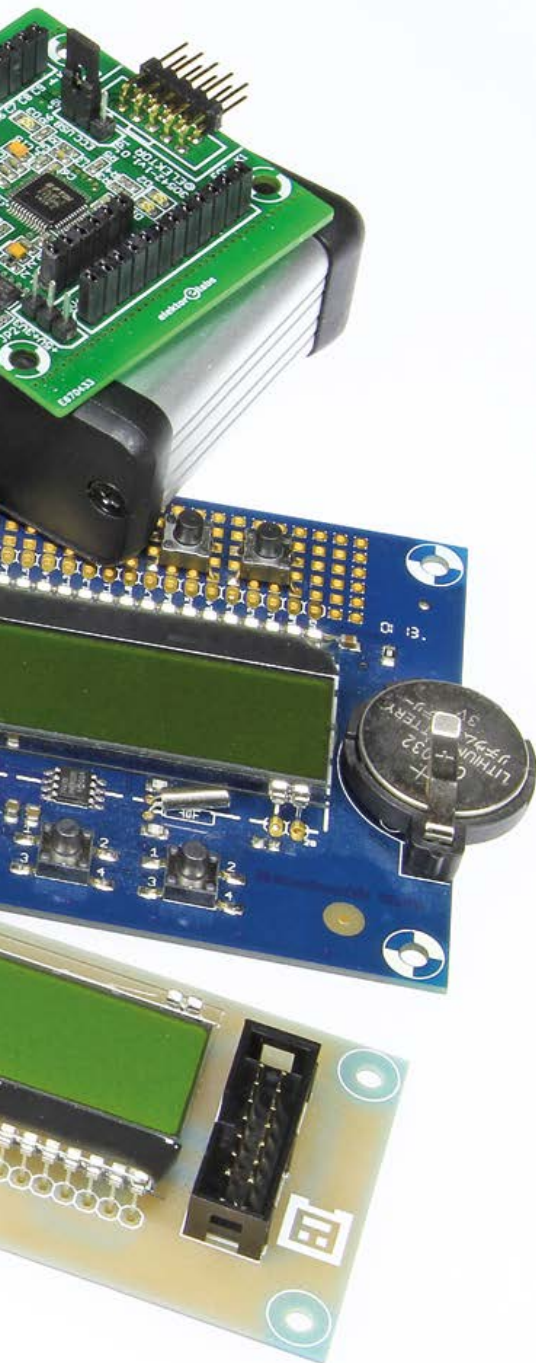


Figure 1. The tables beside the three expansion connectors indicate the connector pin numbering and the respective Port pins. We have included a selection of their logical functions.

▶ Beef up your projects with 32-bit power!

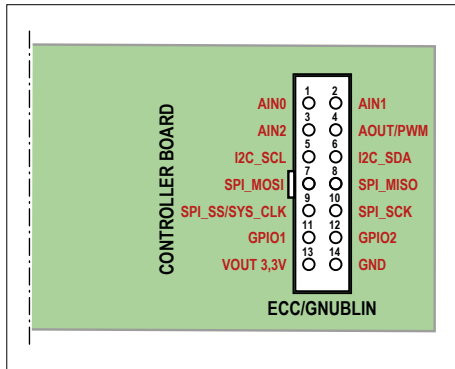


Figure 2. Specification of the 14-pole GnuBlin/EEC connector. When used on a controller board, the polarizing key of the ribbon cable connector points inwards.

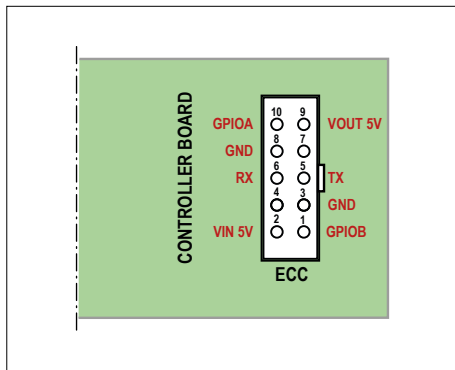


Figure 3. With the ECC UART signals are retransmitted at 5 V level. On a controller board the polarizing key of the ribbon cable connector normally points outwards. If you reverse the connector, the board can then serve as a peripheral for a different controller board or for something such as measurement tests.



Figure 4. Here we have a SAM D20 board and an Arduino Uno (with extension shield) connected for working together. Because the two polarizing keys of the ECC connectors point outwards on each of the boards, we need to use a 'crossover' ribbon cable.

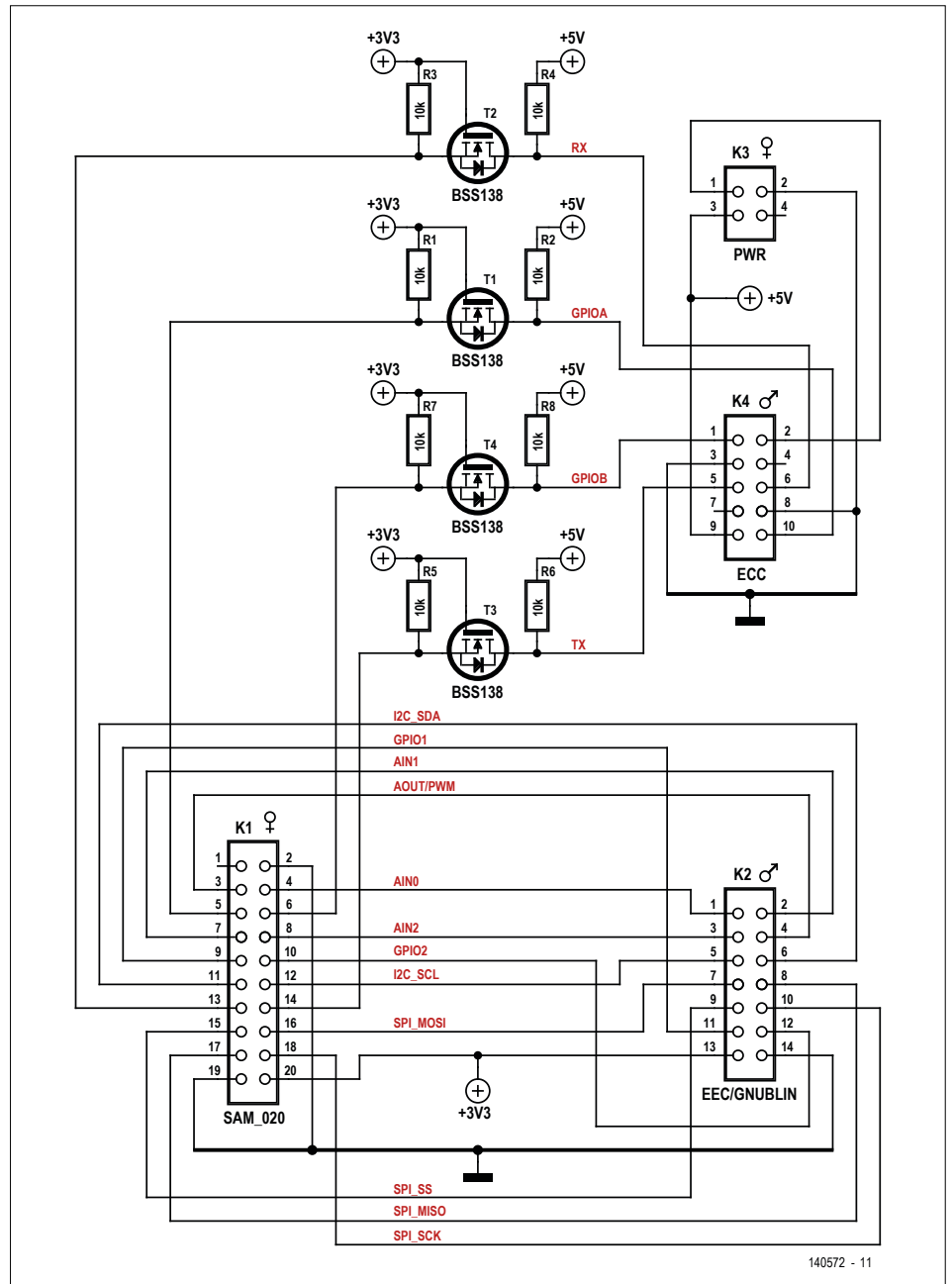


Figure 5. Schematic of the larger adapter with 5 V/3.3 V level conversion for four signals.

Pinout					
These tables indicate the pin numbers of the expansion connectors, the controller pins used (in the example EXT1) and their functions for EEC and ECC.					
		1	2	GND	GND
AOUT	PB0	3	4	PB1	AIN0
GPIOA	PB6	5	6	PB7	GPIOB
AIN1	PB2	7	8	PB3	AIN2
GPIO1	PB4	9	10	PB5	GPIO2
I2C_SDA	PA8	11	12	PA9	I2C_SCL
RX	PB9	13	14	PB8	TX
SPI_SS	PA5	15	16	PA6	SPI_MOSI
SPI_MISO	PA4	17	18	PA7	SPI_SCK
GND	GND	19	20	3,3 V	EEC_3,3

Expansion boards

Together this should provide just about all a developer's heart could desire. Along with some fly-leads and the addition of a breadboard you can already start constructing some simple projects. For creating more demanding prototypes Atmel offers expansion boards that you can hook into. Of course you could even devise a board of your own that features a 2x10 pinheader connector in 90-degree format. However, these boards will then be compatible with only the SAM D20 board and a few similar boards from this manufacturer.

Here is where our two adapter boards come into their own, increasing your scope for rapid prototyping substantially. This is because they enable you to employ a constantly growing menagerie of EEC/ECC expansion boards that for their part work with a broad variety of differing controller boards (Elektor Linux board, Arduino Uno to name just two). PCBs for both adapters are available from the Elektor Shop [6].

So let's begin with the smaller of the two adapters (140572-2). The most significant pins of the Atmel expansion connector are routed to a 14-pin Gnuclin/Embedded Extension Connector (EEC), the specification of which enables the onward transfer of I²C and SPI signals among others (see **Figure 2**). To this you can connect a variety of expansion boards that the team led by Benedikt Sauter has developed for functions such as displays, port expansion, stepping motor control, temperature sensors and the 8-way relay board that we have used already several times. Accompanying these we have Elektor Labs' own 16-bit ADC board and the power meter featured in our last issue (see text box 'EEC modules in Elektor'), with more to follow. By using one or more of these adapter boards, our SAM D20 can now be equipped with various 'extra limbs' to make it usable for the widest variety of measurement and control projects.

Communication

The larger adapter board (140572-1) is wired identically but is provided additionally with an ECC (Embedded Communication Connector) for relaying UART signals (**Figure 3**). At Elektor Labs we have developed all manner of compatible communication modules. Examples of these

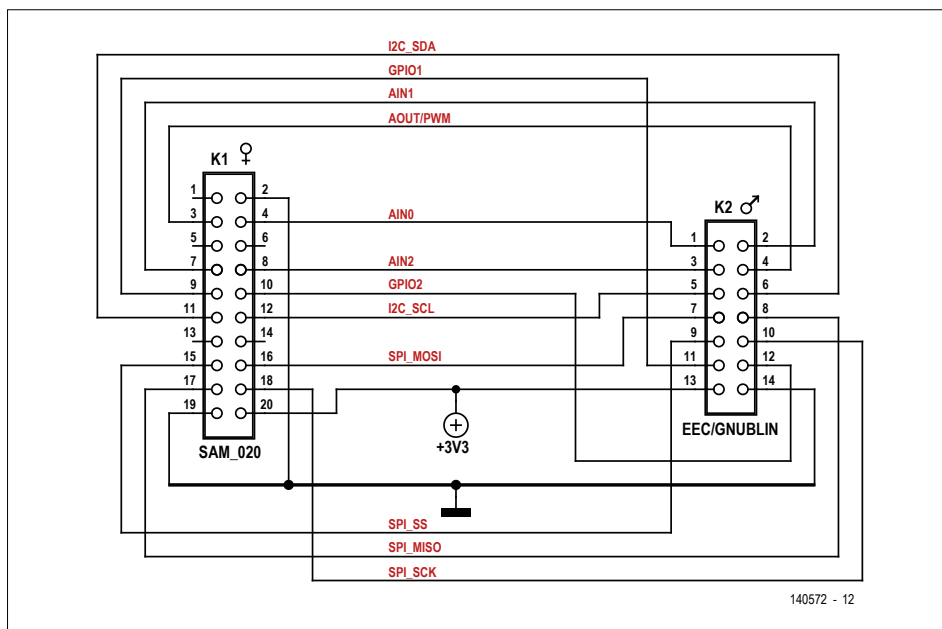


Figure 6. On the smaller adapter only passive signal paths are provided.

include a 433 MHz wireless module, a MIDI in/out module and the NFC gateway for experiments with Near Field Communication (see side panel 'ECC modules in Elektor').

ECC modules (in contrast to EEC modules) operate with 5 V level signals, whereas the SAM D20 works with 3.3 V signals. Consequently four level converters are included on the board (two for RX/TX and two for the digital auxiliary signals GPIOA and GPIOB). Following the ECC

specification, 5 V (V_{out}) has also been provided on one of ECC pins. Fortunately we can take this 5 V from another connector on the ARM board. Another pin of the 4-pole connectors allows 5 V to be fed in for powering the ARM board. The specification of the ECC provides a pin for this purpose, so we can connect the pins just as they are. Now you can power the ARM board from another Controller board or ECC gateway. In principle this would hold good for, say, the Arduino Extension Shield [7]. However, in this case both

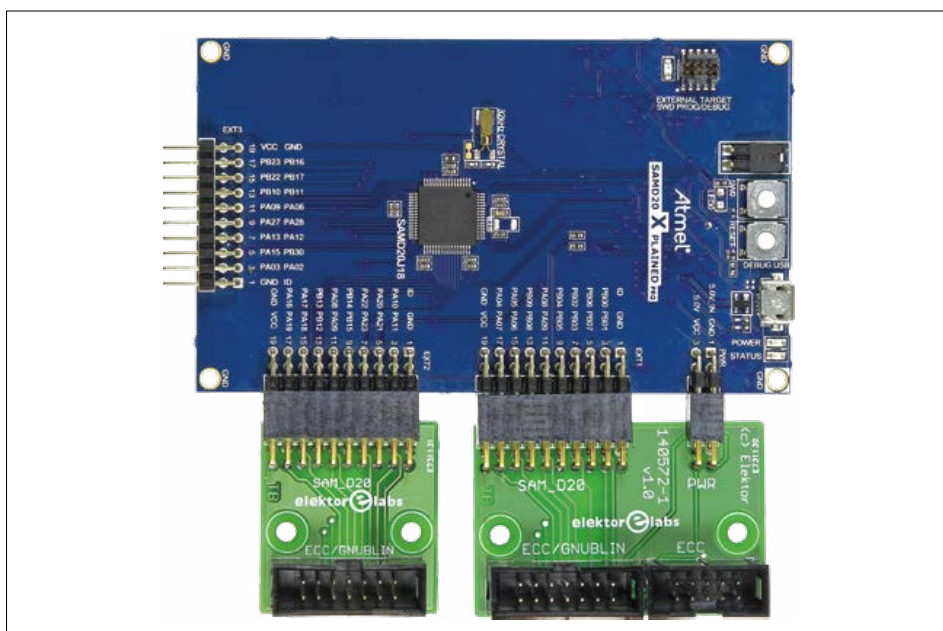


Figure 7. The large adapter board can be connected only to EXT1, whereas there is room to fit all the expansion pins of the small board.

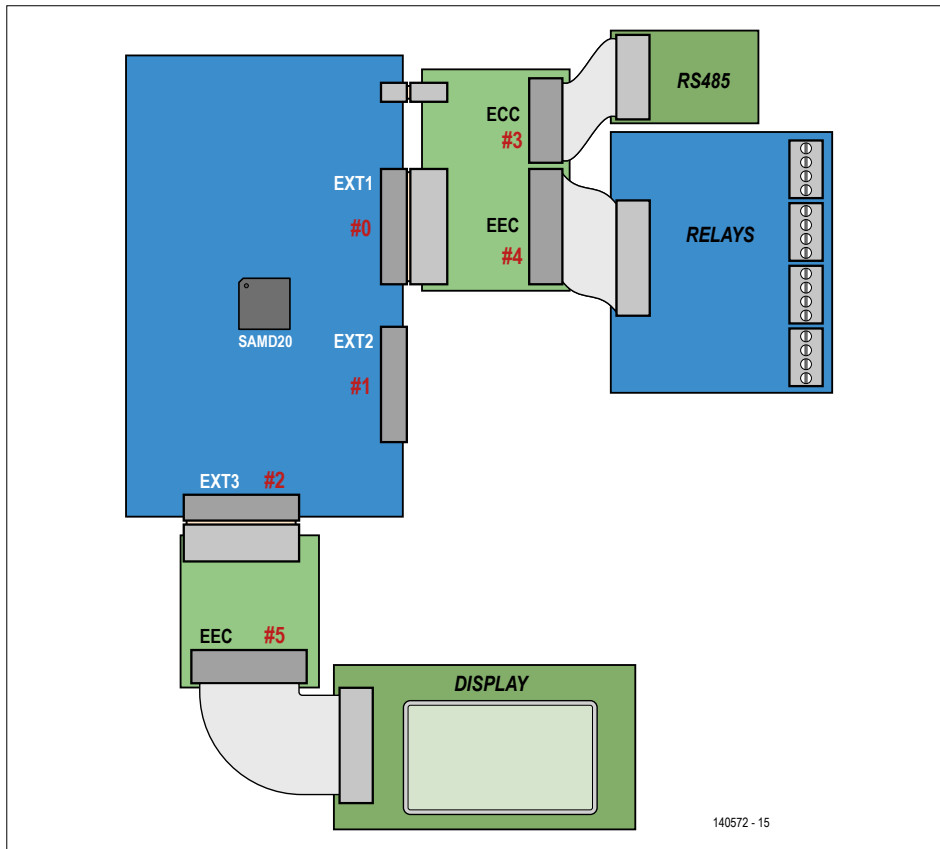


Figure 8. Connector numbering scheme for the demo project. These numbers are assigned when the boards are initialized, starting with the controller board.

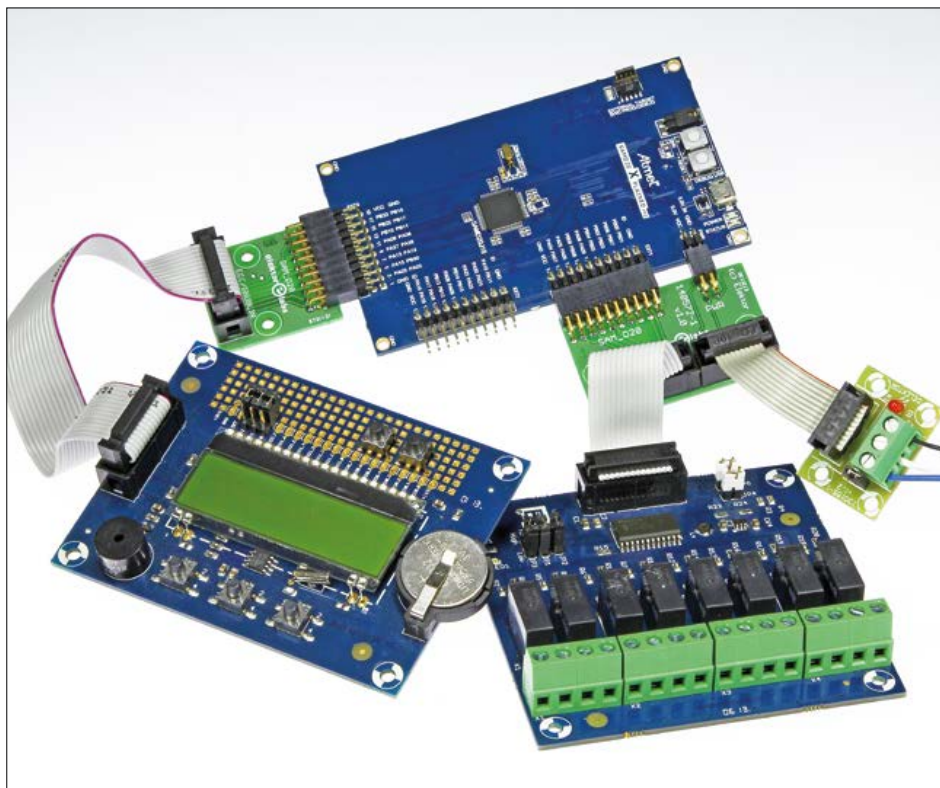


Figure 9. For our demo we connect an RS-485 module to the ECC connector and the relay board to the EEC connector of the large adapter board. A GnuLink Extension Module equipped with a display is wired to the EEC connector of the small adapter board.

of the polarizing keys of the box header connector point outwards, which is an alert that both of these boards perform the same ECC function (i.e. Controller board) and for that reason cannot simply be connected directly using ribbon cable as it comes. However, for this situation you can create a crossover ribbon cable without difficulty. Or else you equip the adapter board with a 10-pin (2x5) pin-header instead of the boxheader and let the polarizing key of the ribbon cable connector point inwards. Now both Controllers can communicate with one another, for instance to collaborate in large-scale measurement projects (**Figure 4**).

Schematics

We don't need to say a lot more about the circuits in **Figure 5/Figure 6**. You will note the purely passive 3.3 V connections along with the four level converters mentioned already. K3 is the small 'PWR' (power) connector at upper right on the ARM board. K1 represents one of the three expansion connectors EXT1, EXT2 or EXT3. The large adapter board, as **Figure 7** clearly shows, can only be connected to EXT1.

In this way the SAM D20 Controller pins that are accessible on the ECC are fixed. They are PB8 and PB9 together with PB7 and PB6 for the auxiliary signals, as you can also check out in the **Pin Table**. This means that on our adapter board only one UART interface of the ARM Controller is usable, which for logical reasons is designated EXT1 in the Atmel software framework. Furthermore, the UART interface available for your own projects is relocated via the EDBG to USB connector [8]. Moving on now to the EEC, when an adapter is connected to EXT1, the five pins provided for digital inputs/outputs and analog inputs are linked to ARM pins PB1 to PB5. The EEC DAC pin (analog output) represents a minor deviation from the original GnuLink specification, which provides a PWM output. If you connect the small adapter board to EXT3, pin PA2 of the SAM D20 is then accessible here; for logical reasons this is the output of the controller's own internal DAC.

The I2C pins SDA and SCL are always linked to PA8 and PA9 (regardless of what the adapter is connected to), since Atmel has routed only this I2C interface to the three expansion connectors on the board. This is not a major disadvantage

of course. We haven't tried this out but it should be possible to control two or more I2C slave modules using the same pins if you configure a different I²C address on each of the slaves.

With the SPI interface it's different. The pins of three different hardware SPI interfaces of the controller are implemented at EXT1 to EXT3. If you install three adapter boards, you can have three fully-functional SPI interfaces on one EEC connector.

Configuration

Very few restrictions are placed upon your rapid prototyping imagination. For instance, without any problems at all you can achieve an ElektorBus control setup with RS-485 connectivity, an SPI display module and the 8-way relay board. If you like, you can revamp the power meter project for ARM or do the same with our NFC demo project, which uses the NFC gateway and a display. Similarly, the MIDI analyzer from our last issue, which receives MIDI data from a keyboard, can now be realized on a 32-bit basis, enabling visions to emerge of some fascinating synthesizer projects.

The NFC demo and the MIDI analyzer projects based on the EFL (Embedded Firmware Library) are particularly quick and easy to adapt to other hardware platforms [9],[10]. In the meantime we have enhanced the EFL hardware layer mentioned a few editions ago for the SAM D20 and the board. The Controller file now supports the two UART interfaces at EDBG and EXT1 together with the three SPI interfaces. The board file makes all three pins of connectors EXT1 to EXT3 accessible.

For the two adapter boards we have produced small extension board files that replicate the wiring in software. If you incorporate these files in your own project, you can use all of the previously created extension board files for ECC and EEC modules just as they are, which of course includes the EFL library files and the majority of the application code, located typically in the main file. Naturally you will need to adapt the initialization of the particular board used with the ApplicationInit function.

Chain-linking

The EFL needs to be told which expansion board is connected to which connector, otherwise the wiring will not match the

Stücklisten

EEC/ECC Board

Resistors

R1-R8 = 10kΩ

Semiconductors

T1-T4 = BSS138

Miscellaneous

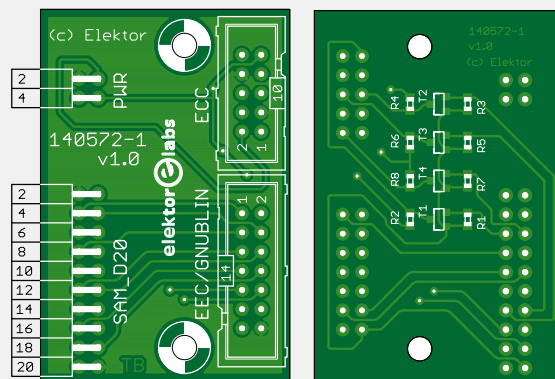
K1 = 20-way (2x10) pinheader receptacle, right-angled, 0.1" pitch

K2 = 14-pin (2x7) boxheader, 0.1" pitch

K3 = 4-way (2x2) pinheader receptacle, right-angled, 0.1" pitch

K4 = 10-pin (2x5) boxheader or 10-pin (2x5) pinheader (see text), 0.1" pitch

PCB # 140572-1 [6]

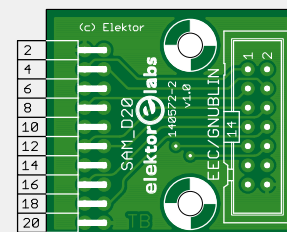


EEC Board

K1 = 20-way (2x10) pinheader receptacle, right-angled, 0.1" pitch

K2 = 14-pin (2x7) boxheader, 0.1" pitch

PCB # 140572-2 [6]



description in the EFL-internal board pin table correctly and the software modules will not interact properly. As with the Triple Arduino Uno, Elektor Extension Shield and ECC/EEC modules, expansion boards can be chained with the EFL without any problems (technically speaking, the Extension Shield is also an adapter board that transposes signals from the familiar Arduino female header to ECC/EEC formats [7]).

We start the Init code with

```
ControllerEFL_SAMD20J18_Init();
```

```
BoardEFL_SAMD20XplainedPro_Init();
```

When the second function is invoked, connectors EXT1 to EXT3 and their pins are registered in the block and board pin tables. From now on the connectors can be accessed using connector block numbers n #0 up to #2, which we have depicted symbolically in **Figure 8**.

We can now for instance connect the large adapter board plus one of the small adapter boards to EXT3. The files *ExtensionEFL-AtmelExt-EECECC.h/.c* plus *ExtensionEFL-AtmelExt-EEC.h/.c* take care of the adapter boards and both files possess an Init function of the corresponding name. These are always accompanied by the block number of the connector that the adapter is connected to, in this case #0 (EXT1) and #2 (EXT3):

```
ExtensionEFL_AtmelExt_EECECC_
  Init(0);
ExtensionEFL_AtmelExt_EEC_Init(2);
```

The result is that when this is invoked later, three further connectors are registered. The upper Init function records in the block table first the ECC connector and then the EEC connector, which are now addressable by the block numbers #3 and #4. Next it's the turn of the lower Init function, which records a further EEC connector (#5).

Initialization of the ECC and EEC boards can now follow. For our demonstration (see below) we connect an RS-485 module to the ECC connector and the relay board to the EEC connector on the large adapter board (see **Figure 9**). The EEC connector on the small adapter board is connected additionally to a GnuBLIN Extension Module that equips our ARM board with a display (controlled using SPI). EFL code files [6] already exist for all of these expansion modules; these files can be integrated into our project exactly as they stand. These too come with corresponding Init functions that we invoke with the appropriate connector block number:

```
ExtensionEFL_ECC_RS485_Init(3);
ExtensionEFL_EEC_Relay8_Init(4);
ExtensionEFL_EEC_ExtensionModule_
  Init(5);
```


This completes our illustration of the way that this remarkably comprehensive system is interconnected. In the process not only six connectors but also two UART interfaces have been entered into the block table to respond using block number n #0 (USB via EDBG) and #1 (RS-485 interface on the ECC). The display, together with the push buttons and the LED on the controller board, all have the block number #0.

Controlling relays

Our demo illustrates relay control using RS-485, which we have featured a number of times previously. At the other end of the RS-485 bus, which we can create with three wires, we connect as always the well-known RS-485/USB converter and behind this a PC. Using a terminal program such as HTerm [11] we can now control the relays directly from the PC. For this we use

```
Relay_LibrarySetup();
UARTInterface_LibrarySetup();
BlockProtocol_
    LibrarySetup(UARTInterface_
        Send, 1,
        UARTInterface_GetRingbuffer(1));
```

in the ApplicationInit function to initialize the Libraries required (among others for the simple Block Protocol) and, by placing the following line of code

```
BlockProtocol_Engine();
```

in the ApplicationLoop function, invoke the Protocol Engine recurrently. This checks whether commands have come in from the terminal program (9600 Baud, 8N1). Using the command

```
R 0 5 +<CR>
```

relay 5 (for example) on the board is activated [12].

In the demo project, which you can download from the Elektor web page for this article [6], a welcome message appears on the display. If you press push button SW0 on the SAM D20 board, the user LED is then toggled.

If you don't have any RS-485 modules or the RS-485/USB adapter, you can alternatively connect the PC direct to the ARM board via USB. In this case you must alter the UART block number, by which block

ECC modules in Elektor Magazine

- RS-485 modules: www.elektormagazine.com/130155
- 433 MHz radio module: www.elektormagazine.com/130023
- NFC gateway: www.elektormagazine.com/140177
- MIDI in/out module: www.elektormagazine.com/150169
- USB/UART converter: www.elektormagazine.com/130542
- Board with 12 capacitive switches: www.elektormagazine.com/130105
- Data logger: www.elektormagazine.com/140126

EEC modules in Elektor Magazine

- 16-bit ADC board: www.elektormagazine.com/130485
- AC/DC power meter: www.elektormagazine.com/140409
- Expansion board with display, port expansion and buzzer: www.elektormagazine.com/120596
- 8-way relay board: www.elektormagazine.com/130157
- Temperature sensor, stepping motor control, real time clock, port expansion: www.elektormagazine.com/130212

protocol communication is processed, from 1 to 0. The corresponding Setup call in the Library should read:

```
BlockProtocol_
    LibrarySetup(UARTInterface_
        Send, 0,
        UARTInterface_GetRingbuffer(0));
```

Now it's all working again! ◀

(140572)



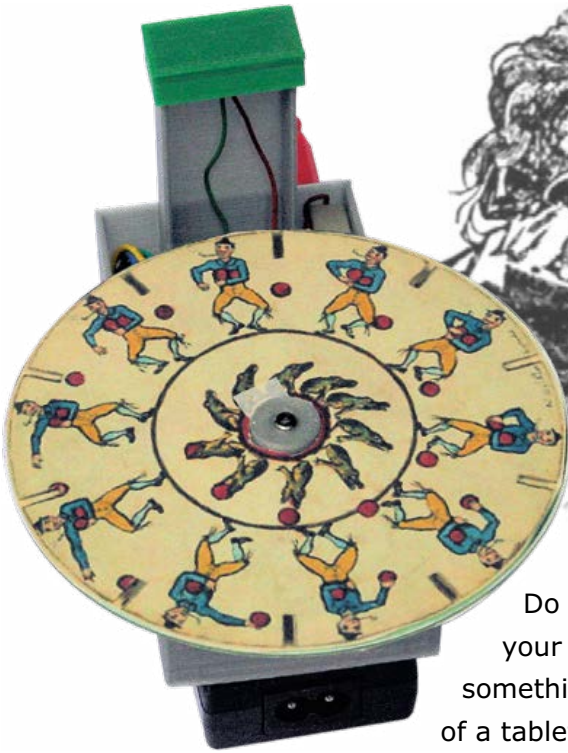
Web Links

- [1] www.elektormagazine.com/140037
- [2] www.elektor.com/samd20 board
- [3] www.atmel.com/Images/Atmel-42129-SAM-D20_Datasheet.pdf
- [4] www.atmel.com/images/atmel-42102-samd20-xplained-pro_user-guide.pdf
- [5] www.atmel.com/tools/atmelstudio.aspx
- [6] www.elektormagazine.com/140572
- [7] www.elektormagazine.com/140009
- [8] www.elektormagazine.com/140512
- [9] www.elektormagazine.com/120668
- [10] www.elektormagazine.com/140328
- [11] www.der-hammer.info/terminal
- [12] www.elektormagazine.com/130154

Digital Zoetrope

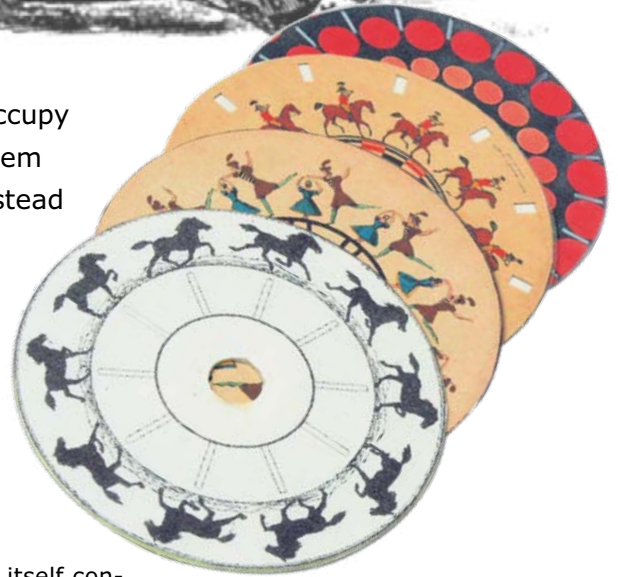
Old & new united

By Willem van Dreumel (Netherlands)



Do you need something to occupy your (grand-)children? Give them something different to gaze at instead of a tablet or some other LCD screen: a 'digital' zoetrope! Perhaps

this term will put you immediately into a nostalgic mood or perhaps it means absolutely nothing to you, but in both cases this 'daedaleum' is a nice project for a dark Sunday afternoon. And for any other day of course!



This article contains a shade of nostalgia. Young people are perhaps no longer familiar with this, but the older generation will likely have come across one of these in the ancient past: the zoetrope [1]. The zoetrope is a kind of predecessor of the Nipkow disk [2]. In a relatively primitive way a picture is put into motion. This makes use of the same principle as film: multiple images are quickly shown one after the other and so create the illusion of a moving image. Because the images are generally on the inside of a revolving cylinder, the scenes are extremely short and are often chosen such that a repetitive motion is shown.

Digital age

The circuit presented here (see **Figure 1**) shows a more contemporary version, where a small stepper motor takes the required number of steps to place the next image in the correct position. By illuminating a bright LED at just the right moment we create an exceptionally stable moving image.

The circuit itself contains only six components. This is of course partly due to the microcontroller, an ATtiny85, which takes care of all the control. The code in the text box **Arduino sketch for the digital zoetrope** ensures that the microcontroller does what we expect from it. We use this to turn the LED on for 1 ms every 20 steps.

You will find comprehensive instructions about the programming of (among others) the ATtiny85 at [3]. Since the software has been written in the form of an Arduino sketch, you can, of course, also use an Arduino board for the control. In this case you will have to pay attention to the five port numbers which are used for the stepper motor and the LED. You will have to change those to suit the Arduino board that you used.

Stepper motor and LED

When using a bipolar stepper motor the control can remain

Arduino sketch for the digital zoetrope [5]

```
//L293D zoetrope
int L0 = 1;//Motor Coil Right
int L1 = 2;
int L3 = 3; //Motor Coil Left
int L4 = 4; //delay between the steps
int flits = 0; //LED
int t = 1; //flash duration
void setup() {
  pinMode(L0, OUTPUT);
  pinMode(L1, OUTPUT);
  pinMode(L3, OUTPUT);
  pinMode(L4, OUTPUT);
  pinMode(flits, OUTPUT);}
void loop(){
  for(int x=0;x<20;x++){ //depends on the step size
    digitalWrite(L3, HIGH); digitalWrite(L4, LOW);
    digitalWrite(L0, HIGH); digitalWrite(L1, LOW);
    delay(t);
    digitalWrite(L3, LOW); digitalWrite(L4, HIGH);
    digitalWrite(L0, LOW); digitalWrite(L1, HIGH);
    delay(t);
    digitalWrite(L3, LOW); digitalWrite(L4, HIGH);
    digitalWrite(L0, HIGH); digitalWrite(L1, LOW);
    delay(t);
    digitalWrite(L3, HIGH); digitalWrite(L4, LOW);
    digitalWrite(L0, LOW); digitalWrite(L1, HIGH);
    delay(t);
    digitalWrite(L3, HIGH); digitalWrite(L4, LOW);
    digitalWrite(L0, HIGH); digitalWrite(L1, LOW);
    delay(1);
    digitalWrite(flits, HIGH);
    delay(1);
    digitalWrite(flits, LOW);
  }
}
```

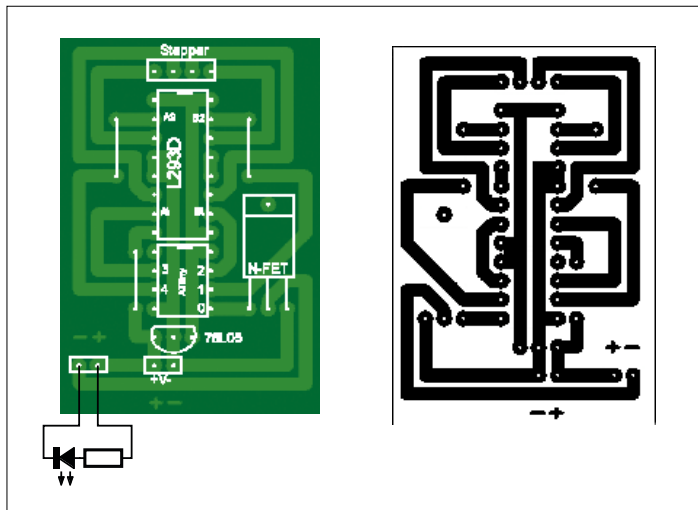


Figure 2. The printed circuit board layout for the digital zoetrope.

To keep everything manageable, the disks can be printed with a diameter of 12 cm so that they can be stuck to an old CD. This is very easily done by pasting the image into a word processing package and setting the image size to 12 cm. Print, cut out and stick... done!

If you fit a small adapter piece to the motor shaft on which you can place the CD, then you can replace the disk while it is rotating and the children are nicely occupied for a rainy Sunday afternoon. Making your own movies from photos or drawings is even nicer of course. When using both sides of the CD for a scene, an extensive collection can be amassed. A nice collection of images for the zoetrope can be found at [4], for example. You can find even more suitable images if you search for 'zoetrope' or 'zoetrope disc'. ◀

(150251)

Web Links

- [1] Zoetrope, Wikipedia entry: <https://en.wikipedia.org/wiki/Zoetrope>
- [2] Nipkow disk, Wikipedia entry: https://en.wikipedia.org/wiki/Nipkow_disk
- [3] Tutorial ATtiny programming: <http://highlowtech.org/?p=1695>
- [4] Zoetrope images collection: www.bekkahwalker.net/MediaArcheology/disks.html
- [5] Sketch: www.elektormagazine.com/150251



ARM CMSIS Developer Competition



The Winners!

Back in March ARM, along with microcontroller manufacturers ST, NXP, Freescale and Infineon, launched a developer competition in conjunction with Elektor. There were dozens of top-quality entries, and judging them was far from easy. And here are the winners!

ARM Cortex-M series microcontrollers have a very wide range of features, and low-level drivers are needed to simplify access to interfaces such as U(S)ARTs, I²C and USB. The CMSIS driver standard is a neat idea by ARM that unifies these drivers in a manufacturer-independent way, and our competition grew out of that idea. Competitors were allowed to choose from a range of four well-equipped development boards produced by ST, NXP, Freescale and Infineon, and were given a free six-month license for the powerful ARM/Keil MDK development environment. A total of US\$10,000 in prizes was up for grabs.

Run on the boards

The demand for boards was tremendous: in total the folks at ARM received around 800 requests, coming from all corners of the world. On the basis of the proposed projects 400 competitors were selected to receive a free board. Developers then had around five months to turn their idea into a reality. The bar was set high: as well as the concept behind the project

being useful and ideally innovative, the software, exploiting the features of the CMSIS driver library, had to be maintainable and portable. Competitors were also encouraged to make the fullest possible use of the features of their selected board.

Choosing the winners

By July dozens of projects had been sent in and it was up to the jury, formed of experts from the manufacturers involved, to make their decision. It was not an easy task! The jury was impressed not only by the high quality of the code and documentation in the entries, but also by the breadth of the range of application areas: from medical electronics to amateur radio and from audio to home automation. And so, without more ado, we announce the winners!

(150297)



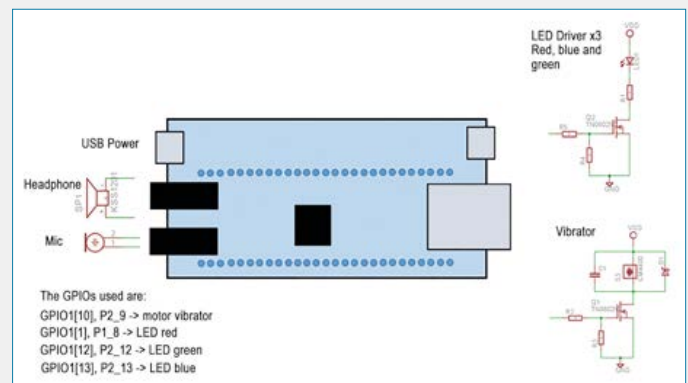
More information on the projects can be found at www.keil.com/contest.

Snore detector with apnea monitor

First prize, US\$5,000

Clemente di Caprio, Rome, Italy

The winning project records snoring sounds during sleep and analyses them for potentially dangerous periods of apnea. The jury was won over by the originality and usefulness of the concept as well as the effective use of software components and development tools. The project is based on an NXP LPC4330, which is a dual-core microcontroller incorporating an ARM Cortex-M4 core and an ARM Cortex-M0 core. The M0 core is responsible for data acquisition and storage, while the M4 core performs the audio analysis functions, taking advantage of its built-in signal-processing instructions. The project uses an MDK-Professional component to implement its file system, and CMSIS drivers to talk to the memory card and the serial and audio interfaces.



WhereSat – a satellite finder for radio amateurs

Second prize, US\$3,000

Stephan Lubbers, Dayton, USA

WhereSat is a portable unit specially designed to help locate radio amateur satellites. These are not in geostationary orbit and therefore must be tracked precisely, which normally requires expensive equipment. WhereSat is a low-cost alternative that works with recorded satellite tracking information. It has inputs from a magnet-

ic compass and an accelerometer for positioning, and LEDs help the user point the antenna in the right direction to find the target satellite. The project is based on an ST Microelectronics STM32F429, an ARM Cortex-M4 microcontroller. It uses the MDK-Professional middleware and the CMSIS drivers for the file system, USB and its convenient graphical user interface.



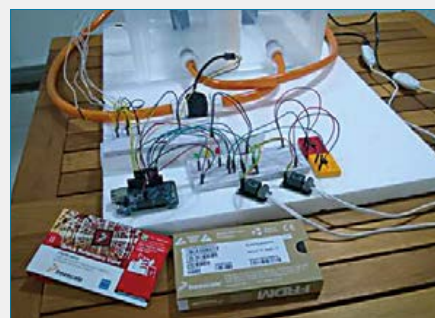
Water usage monitor with web interface

Third prize, US\$1,000

Waldemir Cambiucci, São Paulo, Brazil

This project addresses a pressing problem: water shortages in large Brazilian cities and the consequent rationing of supplies. The solution is very economical, and allows the user to monitor water use in real time and track down possible leaks or wastage. The system processes

the signals from flow sensors and tank level indicators and controls pumps, all under the direction of a web-based interface. The project is based on a Freescale Kinetis K64F microcontroller, which uses the ARM Cortex-M4 core. Interfacing and control functions, as well as access to the Ethernet port, are provided by the CMSIS drivers.



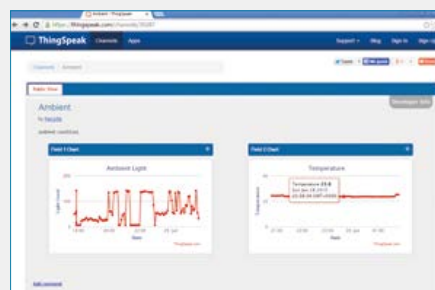
Framework for IoT workshops

Fourth prize, US\$500

Fernando Lichtschein, Buenos Aires, Argentina

This project is an educational platform for IoT devices. It is a high-level tool that helps students to understand how an embedded system and its sensors can be hooked up to the ThingSpeak cloud service provider. The whole thing is built

on the RTX RTOS, which allows additional functions to be implemented easily. The project uses the XMC4500 platform from Infineon and uses the CMSIS drivers for its Internet connection and for connection to sensors over serial ports.



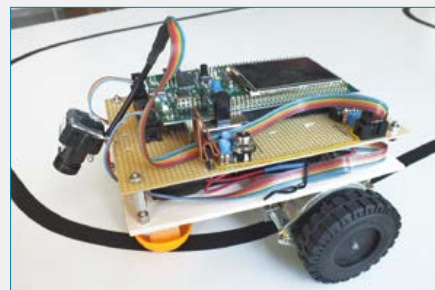
CamBot – with image recognition

Fifth prize, US\$500

Bernhard Schloß, Tübingen, Germany

CamBot is an autonomous three-wheel robot with a camera, which can automatically follow a route marked by a black line. The project is a neat combination of a CMOS camera, stepper motors, and built-in display, and the circuit makes clever use of the pins available on the

STM32 Discovery board. The image processing is done on the Cortex-M4 core in the STM32F429 microcontroller. The CMSIS drivers are used for the I2C and SPI ports which interface to the display and sensors. An MDK-Professional GUI component is responsible for displaying the image.



Honorable mention

- **Nitin Bhaskar:** a smart point-of-sale system using NFC.
- **Stuart Bockman:** a device to measure the jitter of a laser pointer and to detect hand movements using an accelerometer.
- **Javier Fernandez:** a flexible bus analyzer.
- **Alexander Dmitriev:** an intelligent watering system for plants, with a web interface.



welcome in your **ONLINE STORE**

EDITOR'S CHOICE

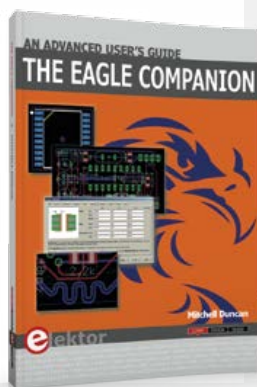


New in the Elektor books portfolio is THE EAGLE COMPANION, An Advanced User's Guide. In over 600 pages, author Mitchell Duncan serves EAGLE users requiring deeper insights in the program's operation, functionality, and advanced features. I honestly thought I knew the basics of EAGLE, but here's a book that explores the more challenging modules, commands and functions which make up the program that rocks the global PCB/CAD scene. Despite PCs & the internet surrounding me 24/7, I like to look up things in a b-o-o-k, especially when faced with a tough PCB design

challenge. Concentration does it, and Mitchell's book is secure on my bookshelf. EAGLE's hidden gem is its User Language Program (ULP) which is a gateway to customizing the program for individual needs and preferences. Mitchell, Cadsoft and I arranged for the full ULP manual to be included in this book in unabridged form.

If ever EAGLE had a faithful companion, it's truly this book. Caution: parcel post delivery!

Jan Buiting, Editor-in-Chief



www.elektor.com/eagle-companion

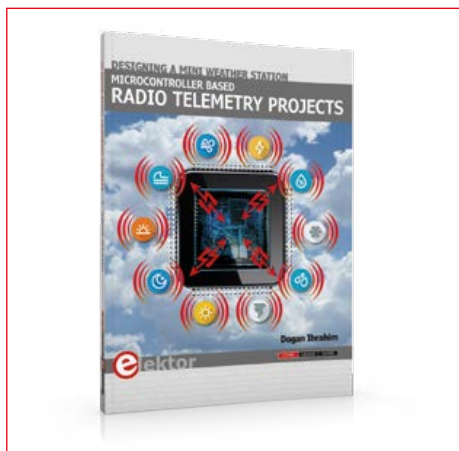
Elektor Bestsellers

1. SmartScope
www.elektor.com/smartscope



2. The EAGLE Companion
www.elektor.com/eagle-companion
3. Red Pitaya
www.elektor.com/red-pitaya
4. VFD Shield for Arduino
www.elektor.com/arduino-vfd-shield
5. C# Programming for Windows and Android
www.elektor.com/c-sharp-book
6. GOLD Membership
www.elektor.com/gold-membership
7. Microcontroller Based Radio Telemetry Projects
www.elektor.com/radio-telemetry

Microcontroller Based Radio Telemetry Projects



This book is written for people who want to learn more about radio telemetry applications and microcontroller programming using the PIC18F series of microcontrollers. The design of a radio telemetry based mini weather station is considered as an example system in the book where the developed system can measure the temperature, humidity, atmospheric pressure, carbon monoxide level, nitrogen dioxide concentration, air quality level, wind direction wind speed and more.

member price: £26.95 • €35.96 • US \$41.00

www.elektor.com/radio-telemetry

C# Programming for Windows and Android



This book is aimed at people who want to learn about the C# language and development environment. It covers steps from installation, the .NET framework and object oriented programming, through to more advanced concepts including database applications, threading and multi-tasking and writing DLLs. The book is based on the Visual Studio 2015 development environment and latest C# additions including WPF applications, LINQ queries, Charts and new commands

member price: £29.95 • €40.46 • US \$46.00

www.elektor.com/c-sharp-book

VFD Shield for Arduino



This Arduino shield contains four Russian IV-3 7-segment VFD tubes with the necessary driver circuits. Four 3mm LEDs provide background lighting for the tubes. Software is available to use this shield in combination with an Arduino Uno as a clock, a counter, or a voltmeter. The design is completely based on through hole components, no SMD components were used. As such, the PCB can easily be assembled by anyone who has some soldering experience.

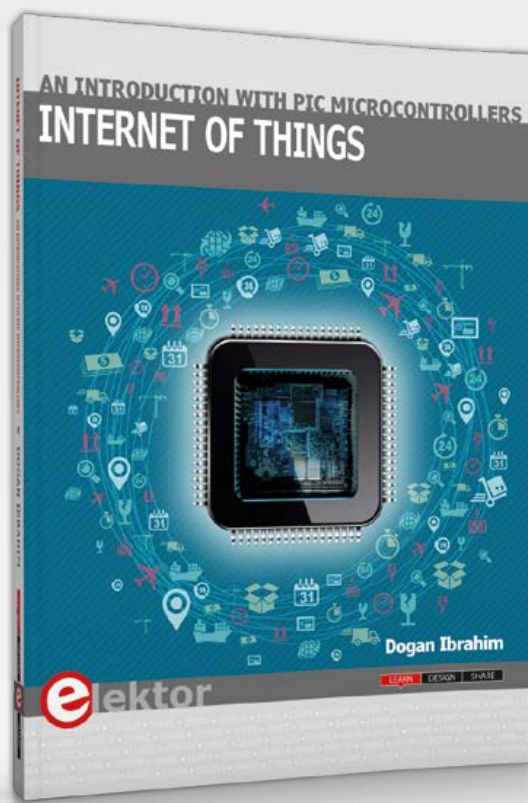
member price: £26.95 • €35.96 • US \$41.00

www.elektor.com/arduino-vfd-shield



Internet of Things

The Internet of Things (IoT) is a new concept in intelligent automation and intelligent monitoring using the Internet as the communications medium. The "Things" in IoT usually refer to devices that have unique identifiers and are connected to the Internet to exchange information with each other. This book is written for students, for practising engineers and for hobbyists who want to learn more about the building blocks of an IoT system and also learn how to setup an IoT system using these blocks. This book has been written with the assumption that the reader has taken a course on digital logic design and has been exposed to writing programs using at least one high-level programming language.



New IoT Book by Dogan Ibrahim

Limited time offer for GREEN and GOLD members:
15% Discount plus free shipping!

PicoScope: a great pocket-sized scope

The PicoScope 2200A Series oscilloscopes: a small, light, modern alternative!

Elektor Store

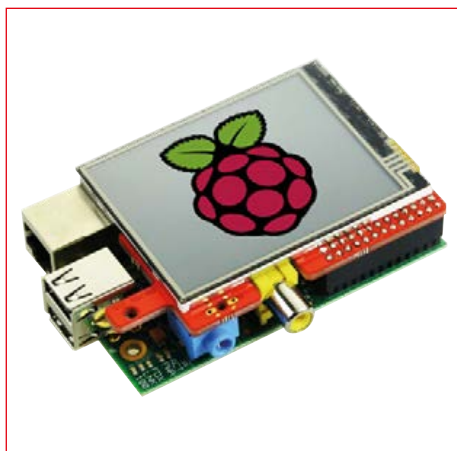
An Aladdin's Cave of books, kits, gizmos and more. Fill your shopping cart today!



MEMBER PRICE: £26.95 • €35.96 • US \$33.00

www.elektor.com/iot-book

Touch-display for Raspberry Pi



From our friends at Watterrott Labor comes the Raspberry Pi Touch Display that's now on offer in our online store. Extend your Raspberry Pi Model B+ or Raspberry Pi 2 with a 2.8" full-color 320x240 pixel display with touch panel. A framebuffer driver is available for the display, allowing use straight out of the box. An extension for your raspberry Pi projects at a very reasonable price!

member price: £22.95 • €31.46 • US \$29.00

www.elektor.com/rpi-touch-display

Vanderveen Trans Tube Amplifiers

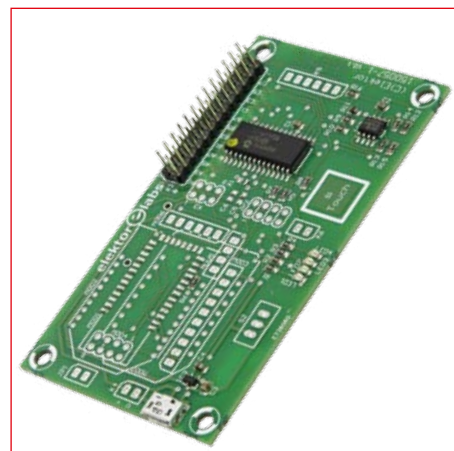


In this book Menno van der Veen describes one of his research projects focusing on the question of whether full compensation for distortion in tubes and output transformers is possible. A variety of methods have been developed with the aim of attaining this goal. One of them has largely been forgotten: transconductance, which means converting current into voltage or voltage into current. Menno van der Veen has breathed new life into this method.

member price: £21.95 • €26.96 • US \$31.00

www.elektor.com/transie

Android I/O Board



The Android I/O Board allows you to control a PIC microcontroller's port pins with your Android device via WiFi, USB or Bluetooth (version 2.0 and 4.0). The specially developed Java classes (libraries) for controlling the Android I/O Board allow you to focus your attention on the application rather than on the handling of complex I/O. Based on a PIC16F1938-I/SO the Android I/O Board offers accessibility to all 25 I/O pins.

member price: £16.95 • €22.46 • US \$26.00

www.elektor.com/android-io-board



By Rei Vilo

I really like the idea of Microduino with a whole ecosystem of two core boards and many shields, available from Day 1! Thanks to the really affordable pricing, instead of picking sensors and spending countless hours on soldering and cabling, just pick the right extension boards you need for a project and stack them for a compact and safe end result.

For legacy Arduino shields, there is a specific extension board that's pin-compatible with the Arduino Uno board. Microduino has enriched the wiki with a whole new section dedicated to training.

Microduino offers compact and stackable boards with many shields, great tutorials, hardware and software that's fully compatible with Arduino. It's perfect for applications where size is at a premium!

Rei Vilo blogs about Embedded Computing embeddedcomputing.weebly.com.

Read this review and more about this product at www.elektor.com/microduino-basic-tutorial-kit



Submit a review of your favorite Elektor product and qualify to win a €100 voucher for redeeming in the Elektor Store.

For further information, please visit www.elektor.com/rotm

HAPPY HALLOWEEN!!

SOUND AND LIGHT EFFECTS FOR YOUR PUMPKIN WITH ELEKTOR'S SCARY HALLOWEEN KIT

FULL KIT
€32.50

WWW.ELEKTOR.COM/HALLOWEEN



PicoScope 2204A

Kit contents and accessories:

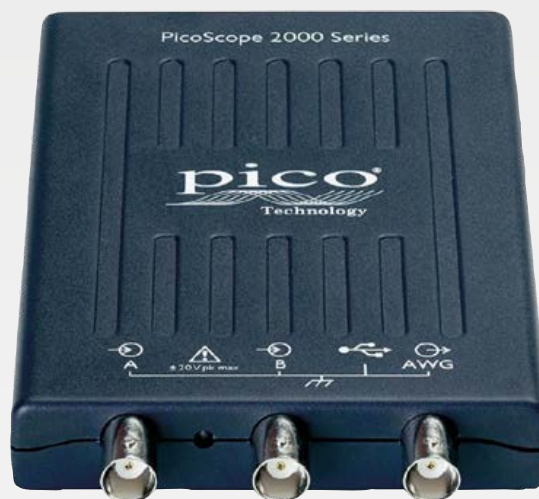
- PicoScope 2204A oscilloscope
- Two x1/x10 passive probes
- USB cable
- Software and reference CD
- Quick Start Guide

The PicoScope 2200A Series oscilloscopes offer a small, light, modern alternative to bulky benchtop devices. You can now fit a 200 MHz, 1 GS/s instrument easily in your laptop bag! They are perfect for engineers on the move, and ideal for a wide range of applications including design, test, education, service, monitoring, fault finding, and repair. A small form factor is not the only benefit of these PC-based scopes. With the PicoScope 6 software, high-end features such as serial decoding and mask limit testing are included as standard. New functionality is regularly delivered through free upgrades, optimized with the help of feedback from customers.



MEMBER PRICE: £103.95 • €143.10 • US \$128.00

www.elektor.com/picoscope



New IoT Book by Dogan Ibrahim

Limited time offer for GREEN and GOLD members:
15% Discount plus free shipping!

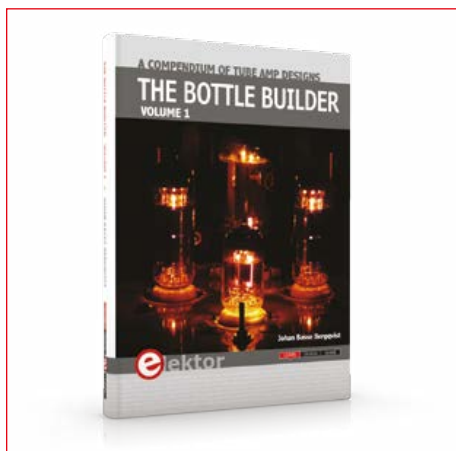
PicoScope: a great pocket-sized scope

The PicoScope 2200A Series oscilloscopes: a small, light, modern alternative!

Elektor Store

An Aladdin's Cave of books, kits, gizmos and more. Fill your shopping cart today!

The Bottle Builder



This massive compendium of tube amplifier designs presents an impressive gallery of tube amplifier projects described not just with a personal stance, but also humor, an open mind, good anecdotes, and an equally good emphasis on all the technical aspects of design and implementation. With every project, the focus is on what makes a particular amplifier "special" in terms of design or performance, which as we all know are not necessarily in agreement.

member price: £52.95 • €71.96 • US \$81.00

www.elektor.com/bottle-builder

Getting Started with the Intel Edison

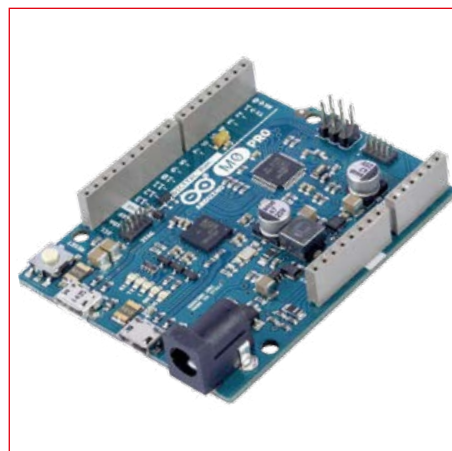


This book is a must have for all those with an active interest in the Internet of Things. 'Getting Started with the Intel Edison' focuses its attention on the Edison, a tiny computer, the size of a postage stamp, with a lot of power and built-in wireless communication capabilities. In 128 pages, renowned author Bert van Dam helps readers get up to speed with the Edison by making it accessible and easy to use. Explore the wonderful world of the Intel Edison now!

member price: £19.95 • €26.96 • US \$31.00

www.elektor.com/edison-book

Arduino M0 Pro



The Arduino M0 pro represents a simple, yet powerful, 32-bit extension of the Arduino UNO platform. The board is powered by Atmel's SAMD21 MCU, featuring a 32-bit ARM Cortex® M0 core. The power of its Atmel's core gives this board an upgraded flexibility and boosts the scope of projects one can think of and make; moreover, it makes the M0 Pro the ideal educational tool for learning about 32-bit application development.

member price: £29.95 • €40.46 • US \$46.00

www.elektor.com/arduino-m0-pro



SHARE

DESIGN

LEARN

Welcome to the **SHARE** section



By **Thijs Beckers** thijs.beckers@eimworld.com

Taking Over

You have perhaps become accustomed to finding the entertaining editorial pieces by our Spanish colleague Jaime in this spot. Unfortunately, I now have to say “ex-colleague”, because Jaime has left Elektor — with a very heavy heart — and now works in Berlin, that great city of culture. He did feel @home at Elektor and got along quite well with practically all of his colleagues. He will therefore, occasionally and from the sideline, work with our team ‘like in the old days’.

His constantly pleasant attitude disguised the forever deadline-taunting delivery times that made our DTP staff’s blood boil at times. At trade shows and events he was always eager to help, during the day he would interact with visitors to the stand in his best possible Dutch, German, English or, of course, Spanish. And in the small hours he would drive the vans back and forth with equipment or quasi as a cab. Outside of work he participated in

mud bathing and other conditioning/bootcamp-ish workouts. His second great hobby was music, over which I have exchanged many a word with him.

We will miss him and wish him “alles Gute”, “all the best”, “todo lo mejor” and much success at his new job!

As of this edition I will govern and introduce the SHARE section of Elektor magazine, although you will likely have realized that already... ◀

(150481)



Rohde & Schwarz HMC8043 PSU



Review and Teardown

A new 3-channel 100-watt power supply — how does it stack up?

By Martin Cooke

First Impressions

Bench power supplies aren't really pieces of kit that get your pulse racing, they just sit there doing their job, in some cases giving gentle background heating to keep the lab warm in the winter and providing enough ballast to stop your bench wobbling about too much. The traditional bench top power supply always came with a sturdy handle on the top to give you a clue that inside is some serious ironwork and that this beast is going to be heavy. But that was all yesterday; buy a power supply today and it doesn't look much different from any other piece of test gear; covered in buttons with a TFT display and the chances are you can wrap your hand around it and pick it up one-handed. That's true of the HMC8043 three channel 100-watt bench power supply from Rohde & Schwarz I received for reviewing.

The front panel layout is restrained and efficient, consistent with their current compact range of test equipment; users of the HMC8012 multimeter will be familiar with the almost identical layout.

The output terminals along the front panel do not include an Earth post. Setting up the functions via the front panel buttons and bright TFT display is fairly intuitive and did not require too much recourse to the operator's manual. Voltage designations on German equipment often use capital *U* instead of *V*. This indicates 'difference' and is best taken to mean *Potential Difference* (*PD*) which is a voltage difference not referenced to ground potential. Most power supplies don't differentiate between *PD*

and *V* but technically *PD* is more accurate and underlines the fact that the outputs of the HMC8043 are fully floating and can withstand a common mode voltage of up to 250 V.

Testing

Turning outputs on/off shows a well behaved output voltage waveform with no tendency to overshoot or undershoot with various loads. The internal processor is monitoring key presses and controls the output appropriately. The output voltage can be adjusted in mV and the current limit in mA or 100 microamps steps for levels less than 1 A. A track option allows all the channels to track together in terms of their voltage and current settings. Each channel of the HMC8043 can be programmed to operate in constant voltage or constant current mode and has an impressive range of protection mechanisms including over-voltage, over-current and over-power protection. Electronic fuse values can also be defined for each channel along with a fuse delay from 10 ms to 10 s to avoid channel dropout at start up inrush. Fusing can be interlinked between channels. In addition to displaying voltage and current readings the instantaneous and accumulated power delivered to the load in W/s in can also be logged. This gives a useful indication of the power requirements and expected battery life particularly if the circuit switches between sleep and active modes.

The three isolated outputs allow the freedom to assign the available 100 W as you wish: connect all three in parallel to give 0-30 V at around 3 A or in series to give 0-99 V at around

1 A output. In this mode it makes sense to set the channels to track so that they share the load. When more than two outputs are connected in series it is possible to exceed the maximum 33 V reverse voltage allowable at the input terminals. This will occur with a load connected, when one of the channels in the series chain turns off due to a low current limiting setting. To ensure the input reverse voltage is not exceeded connect a maximum of two channels only in series.

The most important characteristic of a power supply is the reliability of the output voltage. Working on an expensive prototype you don't want sudden load changes to cause the supply to fluctuate and damage components. Turning outputs on/off using the front panel buttons shows a well behaved output voltage level (**Figure 1**) with no tendency to overshoot or undershoot with various loads. Fan noise was only evident during periods of high power operation. Switched-mode supplies tend to be electrically noisy especially when operating at low load. The HMC8043 showed excellent performance, falling within the 4 mV_{pp} ripple given in the specification.

A situation likely to catch out any microcontroller equipment is a hard turn-off caused by power outage or switch-off at the wall outlet. Unless the raw AC input is monitored closely the microcontroller doesn't get enough warning and the control loop may go unstable as voltage drops. The Rohde & Schwarz HMC8043 reacts to this situation by superimposing an 8-ms (worst case) output spike of around 1.1 V on the output DC level before the voltage falls away (**Figure 2**). This feature may pose a problem for sensitive low voltage circuits; a 3.3-V or 1.8-V system would see its supply peak at around 4.4 V and 2.9 V respectively, and underlines the importance of including on-board regulation on any expensive prototype circuitry to protect against this type of event.

Dynamic-Load Testing

A 0.7-ohm resistor switched in parallel with a 7-ohm resistor was used to test the performance of the supply driving a dynamic load. The resulting DC output level is shown as the upper trace in **Figure 3** with the switching waveform shown as the lower trace. The voltage output is set to 0.700 V and the output current turned up to maximum to ensure that any control mechanism didn't limit the output current. The output characteristics show excellent recovery to the dynamic load with well controlled voltage damping and good phase margin.

The Bob Pease Test

An article written by Bob Pease many years ago told of when he designed a linear regulator chip while working at National Semiconductor. He thought he had taken care of output pro-

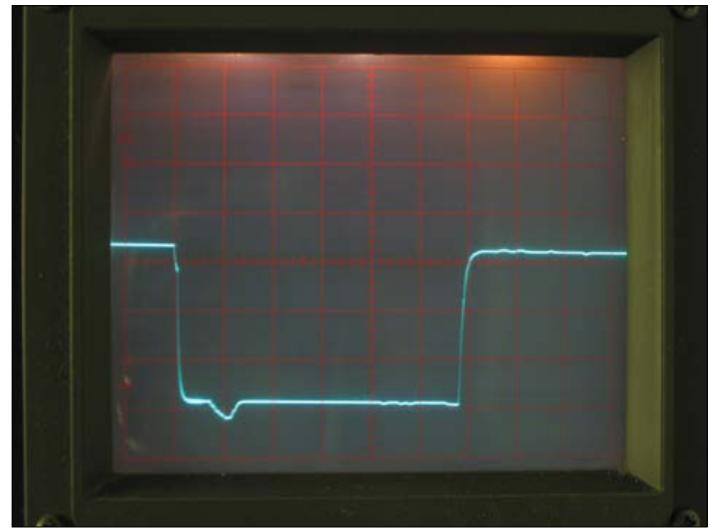


Figure 1. Output voltage 1.800 V soft turn off and on (using front panel push buttons). 500 ms/div, 0.5 V/div. load = 7 ohms with 1.5 microfarads. Captured on a 10 MHz scope.

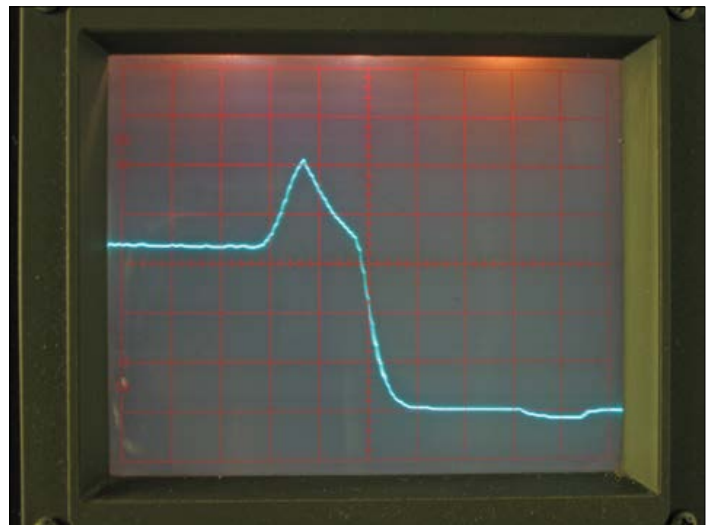


Figure 2. Output voltage 1.800 V hard turn off (power fail) 20 ms/div, 0.5 V/div. load = 7 ohm with + 1.5 microfarads cap (increasing capacitive load to 220 microfarads showed slight improvement (reduced peak)). Captured on a 10 MHz scope.

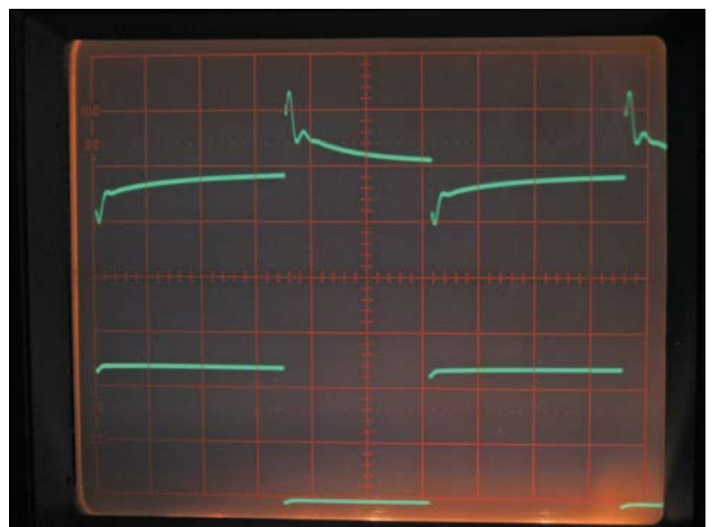


Figure 3. Dynamic Performance.

Timebase 1ms/div 200mV/div (upper trace). 5 V/div (lower trace switching waveform). A load of 0.7 ohms switched in parallel with 7 ohm shows the output takes about 600 microseconds to return within 10% of the final value. It peaks at about +0.26 V and -0.2 V around the nominal value of 0.7 V. The lower waveform is the FET switching signal of the dynamic load. The technical spec indicates 1 ms to get within ± 20 mV, I'm seeing more than 2 ms. Otherwise a well damped control loop.

tection in the design until a co-worker produced a woodworking rasp and asked him to rake it across the outputs. Apparently the intermittent short circuit can defeat the most carefully considered protection and test the regulator control loop to the limit. Sure enough Bob returned to his workplace nursing a smouldering prototype. Lesson learned; could the HMC8043 pass the test? (**Figure 4**) Well there were sparks but in short, yes it did. The same test with three channels in series should not be attempted because the maximum reverse could be exceeded on one channel.

Device Connection Options

The version of the unit supplied is not fitted with an IEEE-488 connector (this is optional) but has a LAN connector for test environments using the newer LXI (LAN Extensions for Instrumentation) network based on Ethernet. The rear mounted USB port allows the unit to be connected to a computer where it can communicate using a virtual COM port or via USB TMC (Test and Measurement Class). Using a virtual COM port allows you to communicate via a standard terminal emulation program



Figure 4. The Pease Test — there were sparks but it survived.

using the SCPI commands after the corresponding Windows drivers have been installed. Alternatively you can use the free Windows application HMExplorer which has a built-in terminal emulator function and also allows you to save screenshots, define an arbitrary waveform and store measurements.

The rear panel connector also allows the connection of an analogue control input voltage. This allows an external analogue voltage in the range from 0 to 10 V (or current from 4 to 20 mA) to control any or all of the three output channels. Assume, for example that channel 1 and 2 have been selected to be controlled by the analogue input. Using the menu the output voltage of channel 1 is set to 30 V and channel 2 to 20 V. Now as the analogue control voltage is increased from zero to 10 V the outputs from channel 1 and 2 will increase proportionally from zero to their maximum setup values.

Useful for development environments using multiple supplies, the sequence option allows turn-on of a channel to be delayed from 10 ms to 10 s after the master On/Off is activated. The analogue control input can also serve as an external trigger input which initiates pre-programmed processes in the power supply. A manual trigger pushbutton is also included on the front panel.

The *EasyArb* feature on the HMC8043 is Rohde & Schwarz's very close approximation of an arbitrary waveform generator (AWG) function. Whereas AWG is usually time-based, in the HMC8043 it can also be event-based, which is exceptional in its price class.

EasyArb allows the output voltage level on any channel to be defined with a resolution of 10 ms, 1 mV and 1 mA. The waveform can be defined using a maximum of 512 points and can be repeated a programmed number of times or indefinitely. Entering the data by hand using the front panel buttons can be tedious; HMExplorer is the more convenient option here.

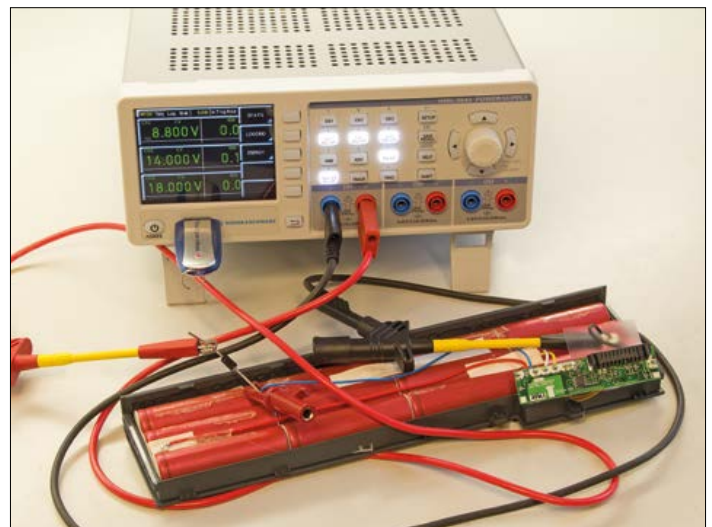


Figure 5. Battery recharging using remote sensing.

Remote Voltage Sensing

The three channel version I tested has a rear socket which amongst other signals, offers remote voltage sensing for all three outputs. This is useful to remove the voltage drop produced in the cables supplying power to the load. This feature can also be handy when the supply is used to recharge a multi-cell rechargeable battery pack. It's always a good idea to include an external diode in series with the battery pack when charging from a power supply. Without the diode, a power outage will often cause the energy stored in the multi-cell battery pack to 'bite back' and destroy the power supply output stage. The remote sense allows the battery pack voltage to be monitored directly, ignoring the non-linear voltage drop induced by the diode (**Figure 5**); the remote sense can compensate for a drop of up to 1 V max.

On this model the sense voltage connection is available on

the rear panel mounted 16-way connector. The special, mating connector is pictured in **Figure 6**. The unit automatically detects the sense input is active and displays shows this on the display. Ensure the remote sense is securely connected before the channel output is enabled; a flaky sense connection can cause the output voltage to fluctuate.

A logging function is also available to allow instantaneous measurements to be stored to the internal memory or a USB memory stick. The measurement interval can be defined in the setup menu. The stored values can be in either .TXT format or .CSV format so that can be input to a spreadsheet like Excel. The really exciting aspect here is that the HMC8043 and the USB stick is all you need to automatically record readings for many hours, even weeks depending only on the size of the USB stick. No remote control by a PC is necessary at any point during the measurements, and the .CSV file loaded from the stick can produce charts almost instantly using Excel. This makes charging, monitoring and condition checking of all sorts of batteries and battery packs a pleasure to do.



Figure 6. The special 16-way connector required for wiring up remote sensing and other applications.

Teardown

Case styling is consistent with other models in Rohde and Schwarz's compact range — in fact without close scrutiny it's quite difficult to tell the difference between the HMC804X power supply and the HMC8012 DMM. The case is a steel sleeve with a perforated upper surface and a polycarbonate end plate forming the front panel. Side vents are preferable for equipment used in general lab environments; there is less likelihood of wire clippings and debris falling into the case and stacked units don't obstruct airflow. For a rack mounted installation this is less of a problem. The case size allows two HMC8043's to be fitted side by side in a 19" rack (a 1-U gap is required above this unit).

Inside the unit (**Figure 7**) is a superb, clean mechanical layout. A 2-mm thick aluminium chassis holds the main double-sided PCB which sits along the bottom of the case. The board uses

SMD components which populate both sides of the board. Behind the front panel is a PCB which interfaces to the display and pushbuttons. The chassis extends up at the rear to form the instrument back panel.

There are no nuts to work loose and cause problems. With the unit open we can see on the upper level, a panel holding a fan and a third-party, industry standard, open-frame 200-watt AC/DC, switched-mode supply unit. This unit supplies 48 V DC to the lower PCB where it divides into three microcontroller (Microchip dsPIC DSC's) controlled DC/DC switching regulators for outputs to the front and rear panels. The PCB quality, component mounting and layout are second to none. With the front panel power switched off the main AC/DC power supply remains continually powered in standby mode until the PSU is turned off at the rear switch. In standby the unit uses just over 8 W.

How does it stack up?

Considering the high build quality and technical sophistication of the HMC8043 it's remarkable that little effort been put into



Figure 7. Inside the case. On top is the Cincon 200-W open frame AC/DC switch mode regulator supplying 48 V to the three microcontroller-controlled DC/DC converters on the lower board.

the instrument documentation. On querying Rohde & Schwarz I was happy to hear though that the issue is being worked on. With all its connectivity options the HMC8043 is perfectly suited to a fully automated test environment but as we saw it also works well as a general purpose bench top power supply for use in the lab. If you are looking for a low-noise bench supply with excellent transient response a traditional standard, heavy, linear design is still hard to beat. If however you want something that can be used in an automated test environment and is programmable, lightweight, versatile and efficient with the ability to control the output to within millivolts and milli-ampères then the superbly built Rohde & Schwarz HMC8043 has plenty to offer. ◀

(150306)

KCS TraceME

2G 3G 4G LBS

LoRa™ BLE M2M

Iridium Sensor



Bluetooth®

iBeacon™

SMS

Glonass GPRS

RF GPS

Internet of Things



LoRa™ Internet of Things

KCS has extended their successful TraceME product line with an advanced module, targeted for worldwide mobility in the Internet of Things era. The latest development of the TraceME GPS/GPRS Track and Trace module will combine the RF location based positioning solution with the LoRa™ technology. This combination offers 'smart objects' being even smarter, since LoRa™ enables long range, battery friendly communication in a wide variety of (M2M) applications. Supporting GPRS/SMS and optional 3G, Wi-Fi, Bluetooth LE, ANT/ANT+ and iBeacon™ provides easy integration with existing wireless networks and mobile apps. Other variants in the high/mid-range and budget-line will follow soon.

ANTI-THEFT module based on RF

KCS TraceME product line offers an intelligent location based positioning solution for indoor and anti-theft applications. The solution is based on RF with an intelligent algorithm of measuring the propagation time of transmitted (proprietary protocol) signals. Unique features are: minimum size (46x21x6.5mm), weight (7 grams for fully equipped PCB) and a standby battery lifespan of more than 10 years. 'Listen before talk' algorithm makes it practically impossible to locate the module, which secures the valuable vehicle or asset. Supporting GPRS/SMS and optional 3G, Wi-Fi, Bluetooth LE, ANT/ANT+ and iBeacon provide easy integration with existing wireless networks and mobile apps.

www.Trace.ME

All trademarks mentioned herein belong to their respective owners.

Rise of the Drones

A few remarkable designs



By Harry Baggen

Quadcopters and their relatives are ragingly popular these days, not only with young people, but many adults as well. Among electronics enthusiasts too, there is much interest in these multi-legged, flying, electronic marvels. They are available in a multitude of types and sizes, ranging from simple toys to large professional versions. In this column we will take a look at a few of the more unusual types.

In bygone times it was the dream of many a boy (and also his father) to fly a model aircraft. Nowadays it appears that everyone wants to play with a quadcopter or something similar. By now we're infested with these things. Toy stores have dirt-cheap models available for children, but also in the higher price categories there is an enormous supply for those who would like to take photos or videos from the air. And then we're not even considering the aircraft for professional users and military purposes.

So drones exist in all price categories, types and sizes. There are a number of remarkable examples regarding their construction or function, and it is a few of these that we will take a look at this time.

Collapsible

Drones — especially the larger models — are quite awkward when considering their size. They are not easily transported because of the four or more protruding motor supports. It is sometimes possible for the propellers to be removed easily, but even then the supports still protrude a considerable way. Some manufacturers have solved this with a construction that includes a collapsible mechanism. Often a kind of umbrella mechanism is used for this, but there are also other ways. The



company Droidworx has developed a quadcopter which looks like a small square black box when folded up [1]. The motor supports in their folded-up state fit exactly in recesses in the housing, and this includes the propellers which can also be folded up. On the front of the enclosure there is a panel with a few membrane pushbuttons and an HD-camera. The 'blu' can be operated using a remote control, a computer or an Android or iOS device. For making recordings of yourself there is a 'selfie' setting where the quadcopter will continue to fly behind you at a distance which you can set. The blu looks rather fragile, at first glance, with its designer looks, but the thing is made from carbon reinforced and nylon parts and as a consequence is very robust. The required computing power is provided by an STM32F7 32-bit microcontroller and a fast GPS receiver is built in by default. The blu costs 500 dollars and there is also a separate Bluetooth controller available in the form of a wrist band.

On a wire

Why would you design a quadcopter that's attached to the end of a cord? Nevertheless that is exactly what has been done with the Fotokite Phi, a project that is collecting funding on Indiegogo to enable production to begin [2]. The Fotokite Phi hovers on a cable behind you and follows you around without the need to control it via a smartphone or remote control. The range is a few meters and the cable holder contains a kind of remote control which can be used to set the height and orientation of the

quadcopter. As a consequence of this design, the quadcopter is very simple to 'fly' and can also be accurately maneuvered when recording indoors. It looks a little bit strange, this 'drone on a leash', but in spite of that, this quadcopter nevertheless incorporates a number of very original ideas. The construction of the thing is very practical, with the folding motor supports that you can unfold in one motion and are locked in place with the turn of a knob. This allows the whole thing to be stored in a tube the size of a thermos flask and is easily carried in a bag. The bracket at the bottom allows a standard GoPro Hero camera to be mounted at various angles.

On the water

The AguaDrone is a quadcopter which was developed for a very novel application. Unfortunately, the crowd-funding campaign on Kickstarter for this device did not collect sufficient money, but the designer nevertheless wants to continue with the production of this remarkable product. It is certainly worthwhile to study this design. The AguaDrone has been specifically designed for anglers. You can let it fly above the water where you intend to fish. The thing has a 'shield' on the bottom, which has a sonar device built in. The AguaDrone lands on the water and uses its sonar to search for fish. If it detects any fish then their location is sent to the smartphone of the angler so that he knows where they are. Subsequently you can let the drone return to you. The sonar shield can be exchanged with a 'fish shield' which has a fishing line and hook attached. Now the AguaDrone returns to the place where the fish were detected and lowers the line into the water. This is indeed a dream come true for any technically gifted fisherman, is it not?

Odd, odd, this 'drone on a leash'



With Intel Edison

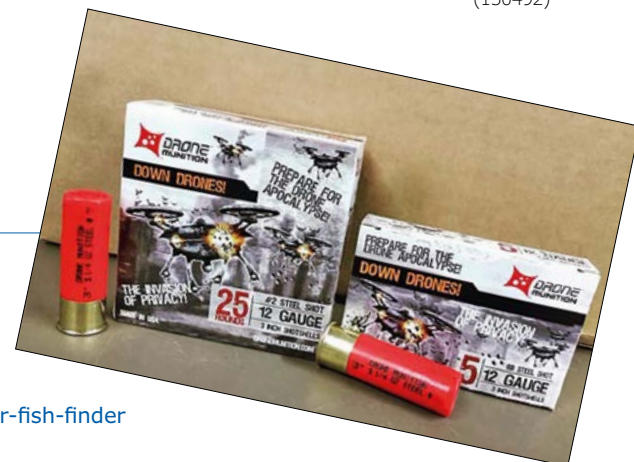
The Eedu (Easy Educational Drone Unit) is a quadcopter which has been especially designed as an educational project [3]. It is easily assembled and disassembled and the electronics has been designed such that it is very easy to experiment with. For this purpose the motherboard does not only contain a fast ARM Cortex M7 microcontroller, but also an Intel Edison module. In addition, the board also has space to accommodate standard Arduino shields. This gives you plenty of freedom when developing your own combinations of hardware, software and applications. Programming is very easy because of the special robot development environment (RDE), Forge. This allows young people and beginners to develop their own programs quickly or adapt existing programs to their desires. Unfortunately, the funding goal for this project on the Kickstarter campaign was not reached either and we now have to wait whether the Eedu will be produced anyway.

Anti-drone

Because of the rapid rise of drones many people feel threatened about their privacy. Particularly in the US this has led to anti-drone products, which can defend against drones or disable them.

For instance, advanced equipment is available to disrupt the wireless communications with the drone. But some people prefer a more radical approach. So there is now already a company which sells special anti-drone shotgun cartridges with shot [5] to shoot the drones from the air... ◀

(150492)

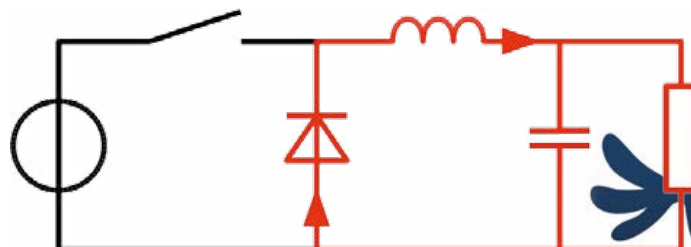
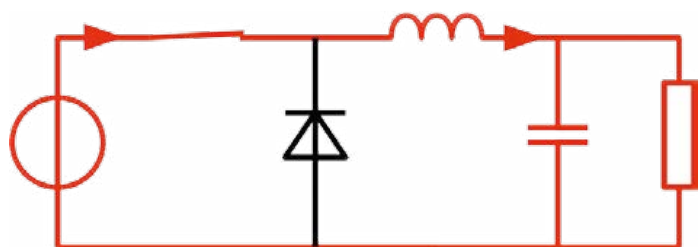


Web Links

- [1] www.mydroidworx.org/blu.html
- [2] www.indiegogo.com/projects/fotokite-phi-tethered-uav-for-aerial-filming
- [3] <http://skyworksas.com/eedu/>
- [4] www.kickstarter.com/projects/1528442751/first-water-proof-drone-with-sonar-fish-finder
- [5] <http://snakerivershootingproducts.com/>

Bullheaded Buck Converter

That's what it normally does!?



By Thijs Beckers (Editor, Elektor Netherlands)

Help! This one we haven't been able to figure out in the lab. So far we failed to get this pigheaded buck-converter under control. What are we doing wrong? We're stuck for ideas...

"With hindsight it is easy to say, but maybe I should not have picked this IC (LM2677T-ADJ, Ed.*)" said a colleague in our lab who was tasked with designing a new laboratory power supply. "Even with a fixed voltage divider for 30 V, a pulsed load of several amps has drastic consequences."

Perhaps we have to take a step back and start from the beginning. The design began with the idea to build a bench supply with the following specifications:

- Adjustable output voltage of 0-30 V
- At least 3 A with current limiting
- Stable under typical lab applications

This is a fairly 'loose' description and it offered the designers a wide choice of potential implementations. And because our name is not Elektor for nothing, and we have not acquired our good reputation with copied circuits and lack of original ideas, we naturally did not choose the easiest and most obvious path. But then you do have the increased likelihood of surprises. As happens to be the case with this circuit.

After some foraging around a few chip manufacturers, looking for interesting possibilities, we came across the LM2677T-ADJ from Texas Instruments. Relatively simple, high efficiency, good voltage regulation and a maximum output current of 5 A all sound ideal. So we set to work.

And soon there was the first schematic, v0.01. We assumed a (reasonably) stable input voltage of 36 V (K1). P-channel JFET T1 prevents switch-on and switch-off artifacts. The feedback input is used for setting the output voltage. By selecting an active control loop (with opamps IC2B and IC2D), we can also implement the current limit at the same time. With IC2A and a shunt resistor (R12) we make the drive circuit for an

ammeter. For IC2A we require a 5-V power supply voltage, this is generated by IC3, an LM317. C4, C5 and R10/C13 are required to make the regulator more stable under load. The time has come for a prototype in hardware.

Without an external load, the output of our prototype had a minimum voltage of about 0.3 V. With a 40-Ω load the lowest adjustable output voltage was about 0.017 V. Not bad, considering the absence of a negative bias voltage. The measured efficiency is also not at all bad at nearly 94% with 30 V/5 A at the output.

The next test, a pulsed load of 5 A with a 50% duty cycle, was less encouraging. The output voltage collapsed like a house of cards and was far from stable. There was no combination of choke and output capacitor for the output filter (L1/C10-C12) to be found that worked properly. Upping to 82 μH instead of 22 μH for L1 changed practically nothing. The only 'advantage' was that the voltage during the pulse load only collapsed and not bounced back to a nasty voltage above the set output voltage (except for a small overshoot). A larger output capacitor had a positive effect on the pulse-load regulation at lower voltages, but caused an even worse drop of output voltage at higher voltages (> 20 V). At about 50% duty cycle the regulation of the controller was also a little more unstable, which has its consequences for the current limit controller.

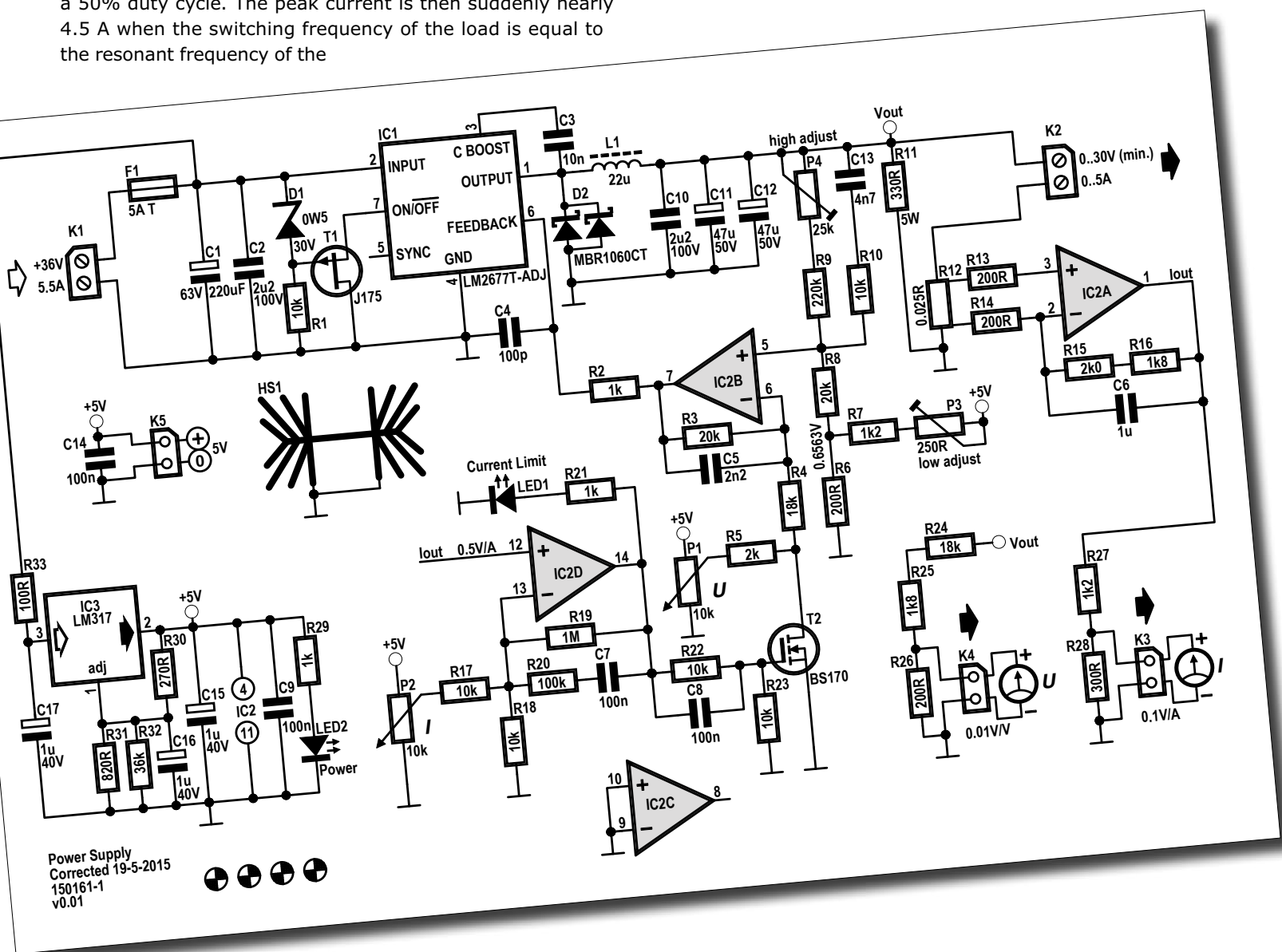
Shorting all the grounds together also gave a slightly better result. A large ground plane on the top is probably essential (the prototype was a single-sided design), but this is unlikely to result in an appreciable improvement in stability. Simulations using Webench (webench.ti.com) result in various combinations of filter components for a range of different output voltages... Perhaps the LM2677 is very sensitive to the ripple at the output caused by the pulse load?



Experimenting with an LC-filter (82 μ H and 220 μ F) connected to a 'normal' DC power supply and a pulsing load resulted in a small surprise: at a 1.32 kHz (approximately the resonant switching frequency) pulse load and 50% duty cycle the DC current through the coils is about equal to half the peak current. Strangely enough this looks to be less on the scope, but a multimeter just indicates half. But there is also an AC-component, at the frequency of the load of course. This was nearly sinusoidal. In our test the combined peak of DC+AC turned out to be one and a half times as big as the maximum current when the load was not switched. That is: if you load a power supply with 3 A continuous, then you would expect an average of 1.5 A when switching the load at a 50% duty cycle. The peak current is then suddenly nearly 4.5 A when the switching frequency of the load is equal to the resonant frequency of the

output filter. The regulator in our power supply would have to be able to supply at least ± 8 A in order to handle a pulse of 5 A without any problems... The internet offered no consolation for this problem. At least, we haven't been able to find anything concrete about it. We are hardly the first to stumble over this?! On the other hand: how realistic is such a pulse load? We can, of course, also limit the power supply at a lower load current, but that would be a shame. At a constant load there is no problem supplying 5 A. Is this typical behavior for all buck-converters with an output filter at their output? Perhaps the output filter is brought into (series) resonance? Because it goes wrong when the switching frequency of the load is in the vicinity of the resonant frequency of the output filter. Perhaps a parasitic current flows somewhere? What do you think of the efficiency? A switching load will, in principle, cause a voltage overshoot, but because the voltage regulator attempts to keep the output voltage constant, the energy stored in the LC circuit can only escape in the form of a current, hence the current peak. Share your opinion with us. Email jan on editor@elektor.com; subject [buck]. To be continued... ❏

(150487)



Dot Labs Is Dee I Wye

And will always be!

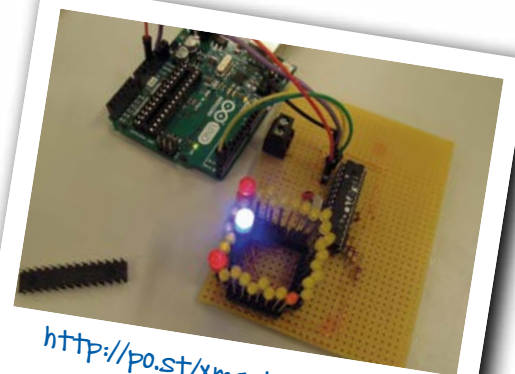


Summer's gone but Elektor.Labs is still on fire! To prove it, here's another compilation of what's new in our user-driven, DIY projects virtual laboratory. Lots of projects, most of them suitable for the average electronics hobbyist, so having reading these pages, go ahead and check out what's new!

Merry Xmas... Eeeerh, Summer Vacation!

Just to make it clear:

We're not yet hibernating just now; just making sure we're in time for the festive season. This Christmas ball consists of a 4x8 LED matrix controlled by an ATmega328P. That makes it easy to program using an Arduino Uno. It's still in the prototype stages, but the PCB and the proper (festive) software will be coming soon!



<http://po.st/xmasball>

Who Wants to be a Millionaire?

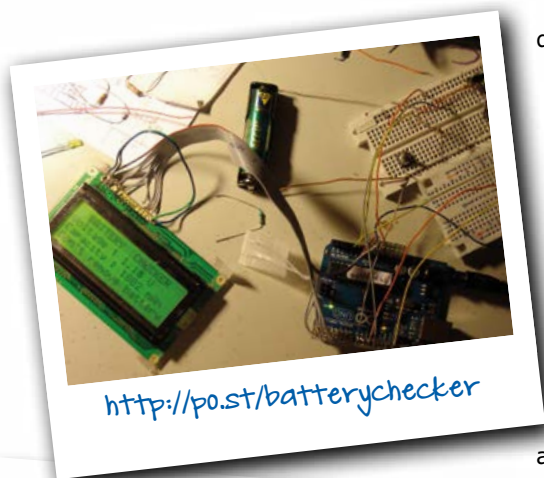
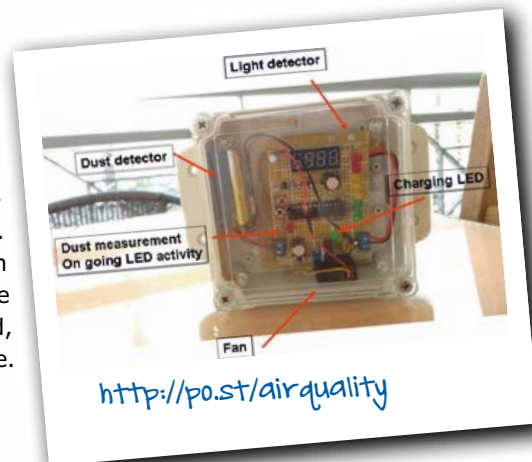
Anyone, right? In any case, to play this one, or any other quiz game properly, you'll want something like this. This set consists of three pushbuttons (transmitters) and a larger module equipped with an LED strip (receiver). The receiver — a recycled spaghetti box — sets the color of the LED strip, according to the button that was first pushed — red, green, or blue — indicating which participant had the answer for the current round, by means of a string sent over-air via Nordic Semiconductor's nRF24L01+ integrated transceivers. Besides, thanks to a proximity sensor, to reset the receiver you only have to wave your hand in front of it. Now you're all set. Wait... the money for the prize? Sorry to say, we cannot help with that...



<http://po.st/quizgame>

The Air that I Breathe

We should be deeply concerned about air pollution, but checking air quality readings online is not always possible, or there are no monitoring stations in our area. So, why not build your own DIY set? This solar-powered system displays the readings taken by a Sharp GP2Y10 IR sensor, and the whole thing is controlled by a PIC18F2420 microcontroller. Besides showing the real-time values in the display, it also uploads them to a server via Wi-Fi, so all the data can be collected, analyzed, and delivered online.

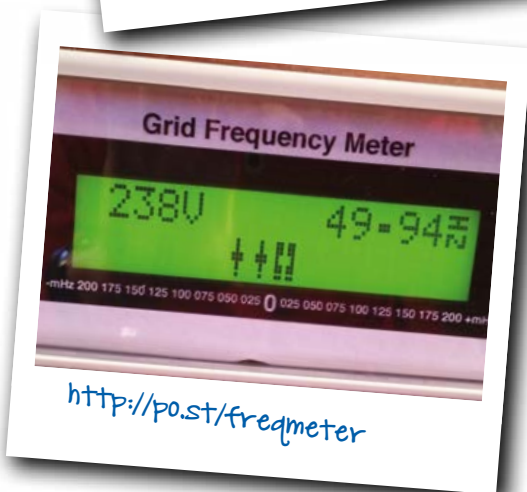


Battery Checker... for Dummies?

This project aims to be an enhanced and simplified version of the "Rechargeable Battery Checker" published in Elektor January 2013, making it easier for the average Joe. Among others, the set of improvements comprises an *Arduinoized* ATmega328, battery detection, a 'start' button, a larger display, and only one fixed discharge current (250 mA). However, the project author is befuddled with the software and the equations, which make the batteries look like three times larger! It's supposed to be a battery checker, not a battery enlarger... So there has to be something wrong! Help please!

Staring at the Freq Meter

A simple but neat project! Based on PICAXE, this utility frequency meter monitors the AC line voltage, the excess and shortage of the available supply in relation to the load, and of course the real-time frequency and any possible drift from the nominal value (up to 200 mHz, no typo). Most people enjoy staring at a fireplace. But EEs are drawn to a nice utility frequency meter. It's impossible to look away!



Your Own UV Exposure Unit

If you think about it, it's just a box with a decent UV light source and a timer, right? It will appear even easier having looked at this manual posted at Dot Labs. This assembly consists on a run-of-the-mill plastic storage box with a 54-pc UV LED matrix, a triple 7 segment display, a buzzer to let us know when the exposure period is over, and of course a PIC microcontroller that governs the entire thing. Because home-made always tastes better!



Behind Every Low-cost Tube Amp is a Low-cost PSU

This high-voltage switched-mode PSU, specifically designed for tube amplifiers, uses a standard 2x30V toroidal transformer, has an output range from 250 V to 350 V, and is capable of providing around 90 watts at 300 mA. Although the complete documentation is in French, the schematics and assembly instructions are pretty straightforward. The simple and neat design makes it quite easy to replicate, so you can spend more time rocking with that brand new DIY tube amp... ◀

(150364)



The Tefifon Grooved-Tape Recorder



Next best thing from Cologne after 4711

By **Peter Beil** (Germany)

As a schoolboy at the beginning of the 1950s (I'm showing my age!) I landed a casual job at the local cinema as a projectionist. In those days music conventionally preceded the main programme, and this was provided by what we would now regard as a bizarre piece of equipment which is the subject of this installment of Retronics.

Probably few people are aware of the fact that between the gramophone record and magnetic tape a different recording medium was in use: a grooved tape.

The 'Tefifon', which used this medium, was originally developed by Dr Karl Daniel (**Figure 1**) in Germany in the 1930s as an answering machine. It used a flexible endless tape loop with parallel grooves that were read (electro-) mechanically (see **Figure 2**). In contrast to later models, this unit was also capable of recording. The project, however, failed to meet the stringent requirements of the German post and telegraph administration.

Arrival of the cassette

At the 1936 radio broadcasting exhibition in Berlin the first 'Tefifon' (with the 'ph' spelling) was demonstrated. Its tape ran at 45.6 cm/s (18 inch/s) and



Figure 1. Dr. Karl Daniel, founder of the Tefi company and eminent Cologne businessman (1905-1977).

Source: H. Jüttemann, 'Das Tefifon'.



Figure 2. The TefiCord answering machine for sure is suitable "for Sound and Speech", yet failed to receive the much coveted German Post approval. Source: Wikipedia.de

offered a massive twelve hours of music playback. After the war, at the beginning of the 1950s, despite the now widespread use of the 78 rpm shellac record, the grooved tape technology still offered some significant advantages. Whereas the records had only three or four minutes of playback time, the post-war Tefifon B 51 model managed over an hour. The tape was 16 mm wide and had about seventy parallel grooves. The playback time allowed complete symphonies, opera and operettas to be played back without interruption: the tracking mechanism moved automatically to the next track, allowing the crystal to move aside in three small steps rather than in one large one. The thin and flexible PVC tape, its four grooves per millimeter foreshadowing the development of microprinting, was no longer prone to creasing or snapping, and the concept of a 'cassette' was also novel. In around 1954 the tape speed was reduced to 19 cm/s (7.5 "/s) and the playback time increased to four hours. Incidentally, modified models also appeared from 1953 that worked with magnetic tape cassettes, but these were never popular because of their high cost compared to 'real' tape players.

Finesses of the Type KC 1

A large number of different models were produced, but easily the most popular was the Tefifon KC 1 player (see **Figure 3**) which cost about 160 Deutschmarks (around €80 or \$85). A big advantage over its predecessors was that the stylus mechanism was located on the top of the unit, presumably for easier servicing.

This model accepted four-hour cassettes and also smaller versions that lasted one hour. When the cassette was mounted on the unit the winding plate in the middle was lifted to reduce friction noise. The tape was drawn out from the middle (**Figure 4**) and the whole loop lies on a driven plate to reduce pulling on the tape reel and prevent it jamming.

It was necessary to pass the tape over a rubber roller by hand — against it was pinched by another roller when the machine was running. The motor turned the pinch roller and its flywheel via a friction drive (**Figure 5a**), while a rubber belt (**Figure 5b**) took power to the winding reel.



Figure 3. Model KC 1 was the Tefi company's first cracker success in the market.



Figure 4. The clever tape jam inhibitor in the Tefifon KC 1 cassette.

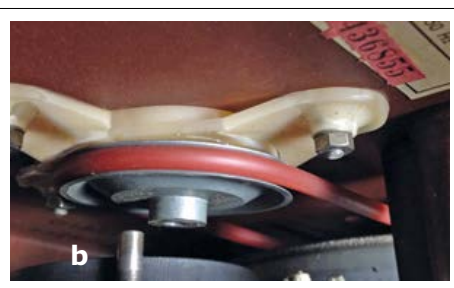
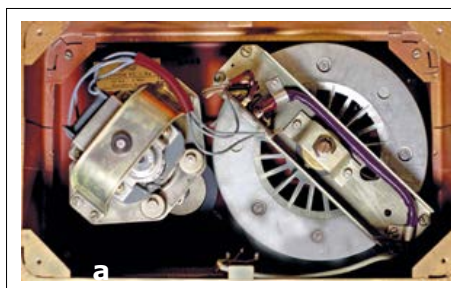


Figure 5. Mechanical assembly of the KC 1 friction drive (5a), and rubber belt and winding reel (5b).



Figure 6. Macro image of the KC 1's pickup stylus.



Figure 7. Stylus landing damper (white arrow).

The pickup stylus took the form of a crystal mounted on a flexible metal beam (**Figure 6**) which in turn was mounted on a rocker mechanism to allow its vertical position to be adjusted. The mechanism was in the shape of a parallelogram with a counterweight which prevented the stylus skating over the tape in the same way that record players can skip tracks. A noteworthy feature is the damping mechanism which allowed a gentle landing on the tape (**Figure 7**) and which, even after sixty years, still works!

Locating Maria Callas' high C

The basic Tefifon KC 1 unit did not include an amplifier, but could instead be connected to any of the radio receivers widely available at the time. Versions with their own built-in radio were made, as well as portable units, in-car units, music centers, TV-combination units, and even a jukebox.

However, the system also had a significant disadvantage: it was not easy to access a desired position on the recording at will. The unit included a reasonably

ESTD 2004

www.elektor.tv



Retronics is a monthly section covering vintage electronics including legendary Elektor designs.

Contributions, suggestions and requests are welcome; please telegraph editor@elektor.com



Figure 8. Futuristic at the time... this bar-graph like tape lapse indicator. *Source: Google images.*

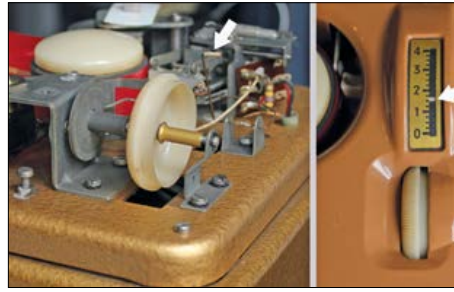


Figure 9. Track selector and indicator mechanisms.

The response was a trick: an adapter cassette with a turntable, powered by a friction drive, along with a retrofitted pickup arm, turned the Tefifon into a record player: see **Figure 10**. To meet the challenge of the single and the LP an 18-minute cassette and a three-minute adapter were designed and sold (**Figure 11**). However, the latter contraption required complex manual intervention (the tension had to be adjusted by hand using a set screw) and the result was not a success.

Graceful degradation

Although it had a stereo output, the Tefifon was unable to keep up with the pace of technological advances. For example, the vinyl record offered a signal-to-noise ratio of 40 to 50 dB; despite technical improvements, the Tefifon always suffered from noise from the tape loop and the drive system, and could only manage a signal-to-noise ratio (S/NR) of around 35 dB. The vinyl record was also superior in terms of wow and flutter.

The arrival of the (Philips' Holland invented) compact cassette (later called 'music cassette') on the market in the mid-1960s finally marked the end of the Tefifon era, although the system survived for some time providing long-play music ('muzak') in stores, bars and cinemas. The Tefi (also spelled: TEFI) factory was closed in 1965 and production ceased, the Neckermann mail-order house taking over what was left of the company. Dr Daniel had left some time previously and had set up under a new name making components for nuclear power stations. The town of Porz (now a district of Cologne) in Germany, where the Tefi factory was located, should name a street in honor of Dr Karl Daniel. Today there is still a large amount of information to be found on the Internet, and many aficionados of vintage audio electronic equipment have restored Tefifon units so that they can relive a little bit of audio history.

(150372)

Note

The author expresses his thanks to the Riedenburg Audio Museum in Bavaria, Germany.



Figure 10. Now you can play records too on your Tefifon!

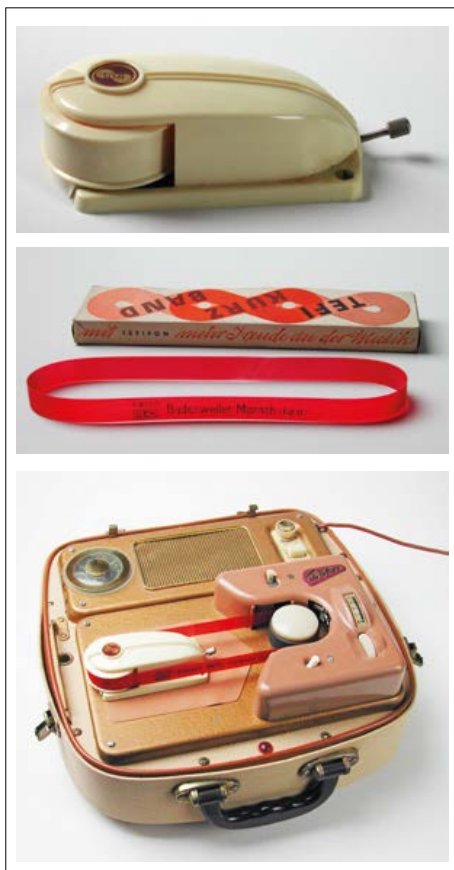


Figure 11. The 3-minute adapter for the Tefifon system. The Ramones would have loved it!

accurate indicator of the position on the pickup arm, but finding exactly the right point on the tape was almost entirely guesswork. In early models there was an illuminated indicator of the arm position (see **Figure 8**), but later models made do with a mechanical pointer.

The track selection wheel (**Figure 9**) was pushed to the right in order to lift the crystal from the tape. The wheel could then be turned to adjust the height of the crystal, its position being indicated on a scale. An alternative was offered by a wired remote control which used a magnet to make the pickup arm jump from track to track.

One other detail: manufacturing the tapes was a very complex affair. At first the grooves were cut; later they were pressed using a die. The difficulty was the join in the tape, which was glued by hand and the grooves subsequently re-pressed. This difficulty was also the reason for abandoning the proposed 'Möbius band' tape (where the tape is given a half-twist before being joined), which would have allowed both sides of the tape to be used.

The turntable add-on

The arrival of the vinyl single on the market heralded competition for the Tefifon system. All the well-known artists were signed up to the big labels, which had no interest in a competing system. The Tefifon system thus had to make do with the B-listers; it was not until the beginning of the 1960s that the Philips repertoire became available. Furthermore, the long playback time of the cassette became a drawback: the hit of the day could be quickly brought on to the market as a single, while on the Tefifon system it was necessary to wait until there was enough material to fill a complete tape.



“Besides instruct, the purpose of Retronics shall be to make engineers smile”



MLX90393

software-defined 3-axis magnetometer



By Clemens Valens

Sensors are primordial to most electronics projects; without them circuits would not (have any) function. Most sensors measure physical quantities that are analogue by nature and using their noise-sensitive signals is not always easy. But digital electronics has come to the rescue and many modern sensors integrate signal processing circuitry and a serial interface. In this article we show you how to use a state-of-the-art tri-axis magnetometer without calculating a single resistor value.

The MLX90393 tri-axis magnetometer is one of the latest magneto products by Belgian sensor specialist Melexis. Another sensor from this manufacturer that you may have heard of is the popular MLX90614 digital infrared thermometer, used in several Elektor projects. The automotive industry is a large consumer of Melexis products and your car probably contains several of them (unless you drive an old-timer).

What is it?

The official product name of the MLX90393 is Triaxis Micropower Magneto-

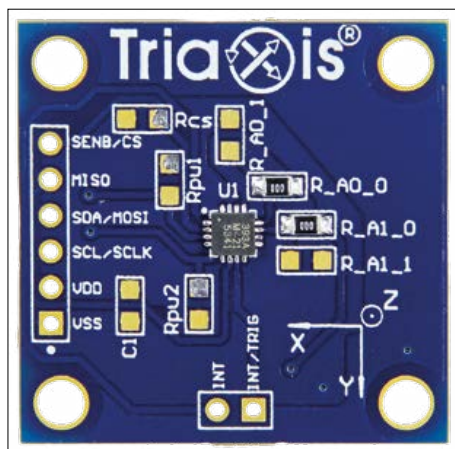
meter and it is part of the Triaxis product family, a family of sensors that use the Hall Effect for measuring changes in magnetic flux density. Other family members are the MLX90333 3D joystick position sensor and the MLX90363 programmable position sensor. Applications of the device range from measuring the Earth's magnetic field (compass) to complex 3D position sensing, gesture recognition and non-contact switching. To make all these applications possible the device's sensitivity is programmable: minimum full-scale equals to about ± 5 mT, the maximum full-scale is around ± 70 mT (values given are

for the X- & Y-plane, they must be doubled for the Z-plane, see inset for units). To compensate for thermal drift the chip includes a thermometer that measures the temperature of the sensor. According to Melexis the MLX90393 is software-defined. We have to take this with a grain of salt because there is no software to be loaded on the device for it to work in a user-defined way; what they mean is that the sensor can be configured by software. Ten 16-bit registers allow you to specify in quite some detail what kind of output you may expect from the sensor.

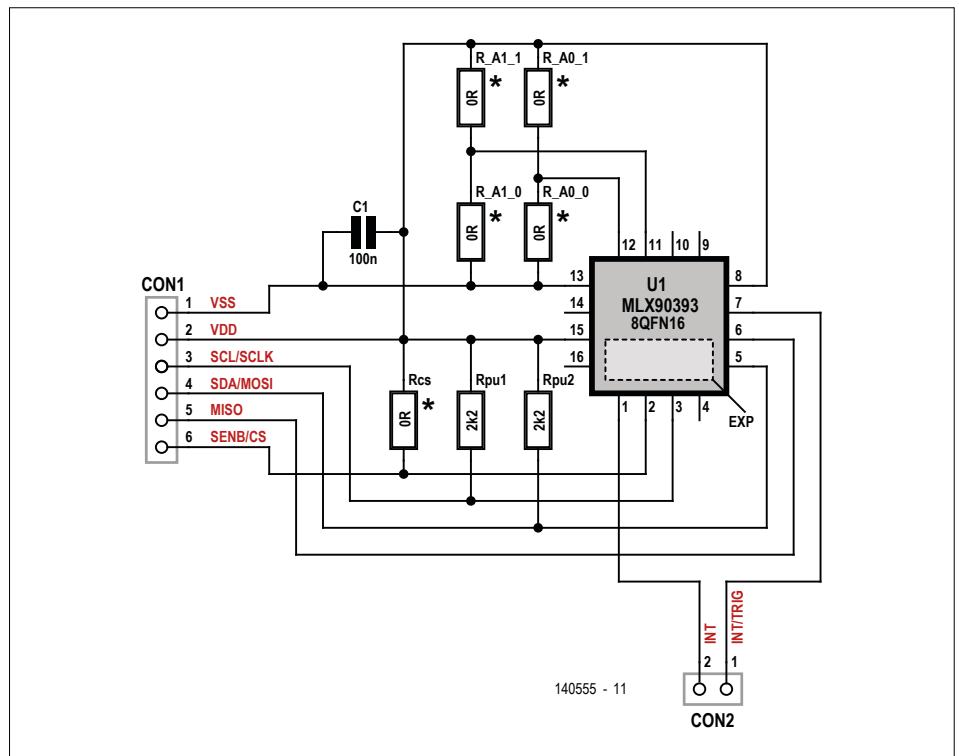
In a similar way the sensor is also hardware-defined because depending on how certain pins are pulled up or down, the device will have an SPI (3-wire and 4-wire modes available, up to 10 MHz) or I²C interface (four different addresses; speeds 10 kHz, 100 kHz and 400 kHz). Reading the datasheet closely it seems that it can even do I²C and SPI at the same time, but it is not clear how. Besides its high configurability, the sensor is also low-current with a nominal operating current specified as 100 μ A.

Using it

The MLX90393 is housed in a tiny 3 x 3 mm 16-pin QFN case, making hand soldering of the chip difficult. Luckily Melexis has designed a handy break-out or evaluation board (EVB) for it (Elektor shop reference 150209-91 [3]) that is easy to use. Also provided are C++ libraries and example code for the LPC1768 mbed module (I²C and SPI), completed by a LabVIEW test application named CherryStone. Download it all from [1]. We wrote an Arduino sketch for the device that communicates with it over the I²C interface (available from [2]). Although the files have a cpp extension our Arduino code is in C making it easy for you to use it with other microcontrollers. The sensor is a 3.3 V device, so interfacing it to an Arduino or any other 5 V microcontroller requires a level shifter, but, if you own a recent Arduino Uno, you can power the module from the Arduino. The mbed module is 3.3 V and can be used directly. To activate the I²C interface you must provide a pull-up resistor on pin 6 to VDD (pin 2) of the module. This can be done



Close up of the MLX90393 evaluation board (EVB).



The schematic of the MLX90393 evaluation board (EVB). Only R_A0_0 and R_A1_0 are mounted, mounting one or more of the others is up to you to decide.

on a breadboard but you can also solder it directly on the PCB at the position of Rcs. The I²C address is preconfigured to 0x0c but you can change it by relocating the resistors on R_Ax_y. Depending on your setup you may need to add the I²C pull-up resistors (Rpu_x). The mbed I²C example expects that you connect the mbed pins 21 and 22 to the I²C address lines, but they are not accessible on the EVB's connector. Leaving them unconnected should be OK as the firmware defaults to zero for these two lines (at least that is what Melexis told us).

Once the hardware wired up correctly – SPI or I²C, mbed, Arduino or something else – you can start playing with the MLX90393. To make it do something you must send it commands. You start by configuring the sensor, either “manually” or by loading a configuration that you previously stored in the chip’s internal non-volatile memory. The next step is to choose the operating mode. There are three: single, burst and wake-up on change (WOC). In single (measurement) mode the device takes one measurement when the host asks for one. In

Gauss, Tesla, Weber, Henry, Maxwell...

For those who suffer from sleep onset insomnia it may be useful at night in bed while drinking a hot cup of cocoa (keeps Alzheimer away, it appears) to review the units that are used for magnetic quantities.

Magnetic flux density or magnetic induction, or simply magnetic fields, used to be measured in gauss (G), but in the SI system this unit was replaced by the tesla (T). One gauss is defined as one maxwell per square centimeter (Mx/cm²). The maxwell is the pre-SI unit for magnetic flux (previously called a *line*) and has been replaced by the weber (wb). 1 Mx = 10⁻⁸ Wb, so 1 G = 10⁻⁸ Wb/cm². 1 G is also equal 100 μ T and so 1 T = 1 Wb/m² (note the switch from cm² to m²). But, since T is equal to V·s/m², one gauss is equal to 10⁻⁴ kg/(C·s). Because the tesla is quite large to express the Earth’s magnetic field and other natural phenomena in, geophysicists tend to use the gamma (γ), where 1 γ = 1 nT. The henry (H) is the unit for inductance, closely related to magnetism: H = Wb/A = T·m²/A = Ω ·s. If you made it this far, you should consider consulting a sleep specialist.

Hall plate spinning?

The datasheet of the MLX90393 contains references to Hall plate spinning and chopping. Since you can set values for spinning and chopping of the Hall plates you might as well know what this is about. The Hall plate is actually the magnetic field sensor element and it is called a plate because of its plate-like shape. Spinning it does not mean that it rotates, but that the drive voltage contacts and the output contacts are permuted periodically. This is possible because the plate is symmetric. The goal is to reduce the sensor's offset. Chopping refers to the well-known chopping amplifier

that also reduces offset. The more you spin and the more you chop, the better the result. On the other hand, lots of spin & chop increases measurement time and energy consumption, so you need a compromise. Sticking with the default values is probably good enough for most users.

Want to know more? Ask the competitor: www.allegromicro.com/en/Design-Center/Technical-Documents/Hall-Effect-Sensor-IC-Publications/Monolithic-Magnetic-Hall-Sensor-ICs-Using-Dynamic-Quadrature-Offset-Cancellation.aspx

burst mode the chip continuously takes measurements at a programmable rate and sets the ready flag (pin) every time a new sample is available. WOC mode is like burst mode, except that the flag is only hoisted when a value passes a previously set threshold. In all modes the host is responsible for collecting the acquired sensor data by issuing read commands. When you select the mode, you must also tell the chip which axis (X, Y, Z and Temperature, in the documentation referred to as "ZYXT") you want to measure. Any combination is possible.

It is recommended (although officially not necessary) to send the Exit Mode (EX) command every time you change operating mode.

Commands are acknowledged with a status byte and optional data. You must check the status byte to find out if all is well and how much data bytes you are supposed to read.

As briefly mentioned before, the MLX90393 has an internal EEPROM. This non-volatile memory is divided into several sections, one of which is available to

the user as free-form memory. You can store phone numbers here if you like. Refer to the MLX90393 Getting Started Guide [1] for the meaning of all the bit fields, commands and sensor parameters.

CherryStone

A LabVIEW application is available on the Melexis website that you can use to evaluate the EVB with an mbed LPC1768 module. The wiring of the circuit is contained in the filename of the firmware that must be loaded onto the mbed module. Because CherryStone is a LabVIEW application, users who do not have LabVIEW must install the LabVIEW Run-Time Engine 2012 (32 bit) first before installing NI-VISA 5.0.3. These programs are available from the National Instruments website. Unfortunately this does not work on Windows 8 and up, and not on Linux either. OSX may work. Full details can be found in the Getting Started guide mentioned above.

To select the I²C interface the SPI/I2C button must not "light" up. ◀

(140555)



Screenshot of the CherryStone test program.

Links

- [1] www.melexis.com/MLX90393
- [2] www.elektormagazine.com/140555
- [3] www.elektor.com/mlx90393-triaxis-micropower-magnetometer

Advertisement



HAMMOND
MANUFACTURING®

Housings for Raspberry Pi, Arduino and many other bareboard computers

- Enclosure
- Platform

+ 44 1256 812812

sales@hammondmfg.eu /1593HAM.htm



The choice is yours.

- Enclosure for all round protection
- Platform for all round access
- Design-specific versions for all popular models
- Visit hammondmfg.com for full details



/1593HAMEGG.htm

Coming soon: is it e-ethical?

By **Tessel Renzenbrink**

“Show of hands, please: which one of you thinks electronic companies should play an active role in determining the electronic ethics of the future?”

This question was asked during an annual conference of a national electronics branch organization. How many hands from the 300 electronic entrepreneurs in the congress room were raised, you think? Just five.

So a whopping 98% of businessmen and decision makers believe they can follow ostrich policy when it comes to questions like: “Should we let drivers program their car for Yes or No automatic braking for a dog in a situation that’s dangerous to themselves?” Elektor believes that has to change!

At least 290 hands should respond YES next time. Because we, developers and producers, bear a substantial responsibility when it comes to electronics-related ethics. Not forgetting it implies great business too. All ethical discussions will lead to innovation one way or another, to new opportunities, new business, and, last but not least, it’s plain fun to discuss how electronics-driven innovation can change the world.

That is why we launch Elektor Ethics from November. Online and in print. It will take the place of our Elektor World section, which will continue to be published regularly, but not in a fixed format. Do you have ethical questions or issues? Contact me at tessel.renzenbrink@eimworld.com. I am dedicated to coordinating the discussions about ethics in this space.

(150491)



Hexadoku The Original Elektorized Sudoku

Time flies when you're having fun. 2015 is almost behind us and here's the final Hexadoku for this year. It should keep you busy for a good eight weeks. Find the solution in the gray boxes, submit it to us by email, and you automatically enter the prize draw for one of three Elektor book vouchers.

The Hexadoku puzzle employs numbers in 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 prize draw. All you need to do is send us **the numbers in the gray boxes**.



Solve Hexadoku and win!

Correct solutions received from the entire Elektor readership automatically enter a prize draw for three Elektor Book Vouchers worth **\$70.00 / £40.00 / €50.00 each**, which should encourage all Elektor readers to participate.

Participate!

Ultimately **December 1, 2015**, supply your name, street address and the solution (the numbers in the gray boxes) by email to: hexadoku@elektor.com

Prize winners

The solution of Hexadoku installment 5/2015 (September & October) is: **F1B94**.

The €50 / £40 / \$70 book vouchers have been awarded to: Chris Klaassen (Netherlands); Doug Hicks (Canada); Bart Grefte (Netherlands); Brian Wood (UK); Mike Corteling (Australia).

Congratulations everyone!

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

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

The competition is not open to employees of Elektor International Media, its subsidiaries, licensees and/or associated publishing houses.

Do you need Digital Power with next-generation capabilities?

New dsPIC® DSCs set benchmarks for size, latency and power consumption



Enabling sophisticated control algorithms operating at higher switching frequencies and Live Update Flash, Microchip's 16-bit dsPIC33EP "GS" Digital Signal Controllers offer next-generation digital-power performance.

These DSCs consume up to 80% less power in any application and provide less than half the latency of the previous generation when used in a three-pole three-zero compensator.

In addition to exceptional performance for non-linear, predictive and adaptive control algorithms, the DSPIC33EP "GS" family offers higher integration and more features in packages which include the industry's smallest digital-power-optimised DSC, 4 x 4 mm UQFN.



microchip
DIRECT
www.microchipdirect.com

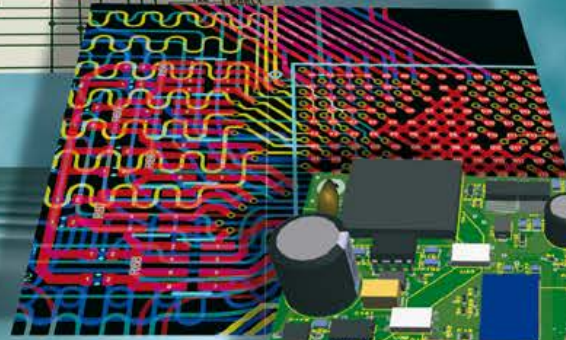
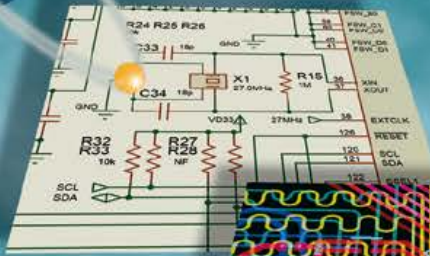
 **MICROCHIP**

www.microchip.com/get/eudspic33ep

PROTEUS 8.3

ECAD to MCAD made easy

Data Exchange with STEP/IGES



AUTODESK. PTC
SOLIDWORKS

The Proteus Design Suite now includes full support for data exchange with Mechanical CAD packages via the STEP/IGES file formats. This allows you to better visualise your design and helps quickly solve fixtures, fittings and casement problems.

Import 3D STEP/IGES models for your parts and visualise inside the Proteus Design Suite. Export your completed board to Solidworks or other MCAD software.

Visit www.labcenter.com
Tel: +44 01756753440 E-Mail info@labcenter.com