www.elektor-magazine.com

.magazine

September 2014

elektor

Professional Lab Supply

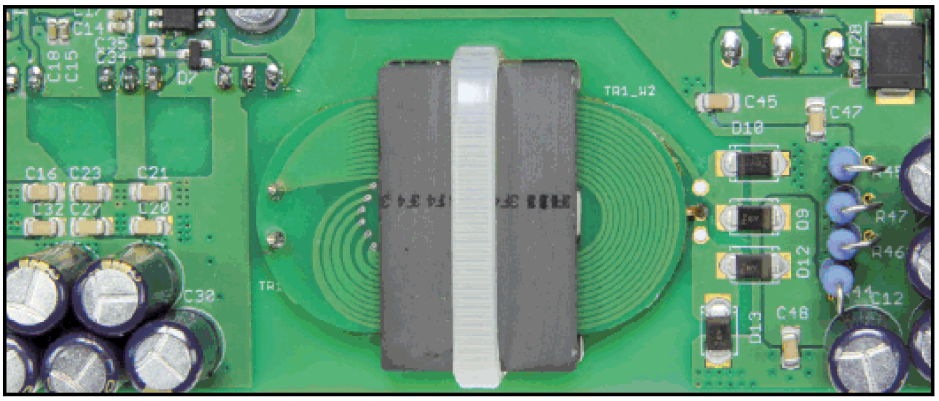
Superb stability
Accurate control & response

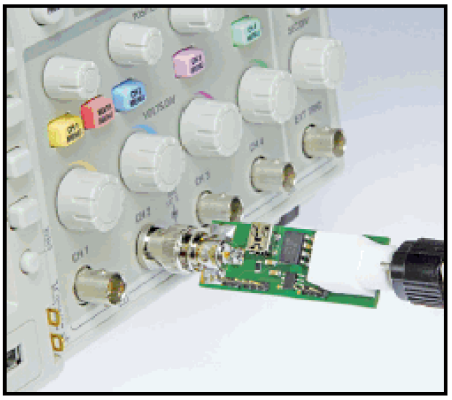
- **T-Boards 8/14/28** | Microcontroller BootCamp (5) | 16-Bit Data Logger
- Isolated Oscilloscope Probe | **Three-way CH Boiler Valve Monitor** | Efficient Water
- Solenoid Valve | **DesignSpark Tips & Tricks** | Weird Components: Peltier Modules
- **ELPP: Elektor Labs Preferred Parts** | Inexpensive MyDAQ
- Connectivity **NI VirtualBench Review** | Strain Gage with PSoC
- **Retronics: BK560 Programmable IC Tester**

US $9.00 - Canada $10.00

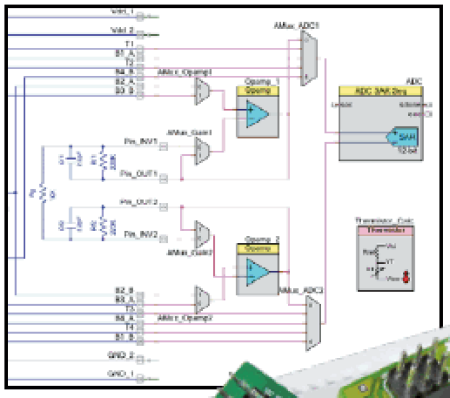
09

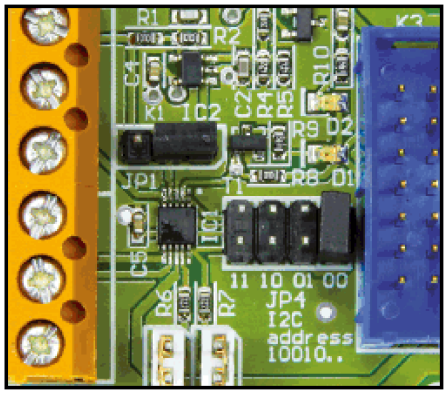
elektor magazine

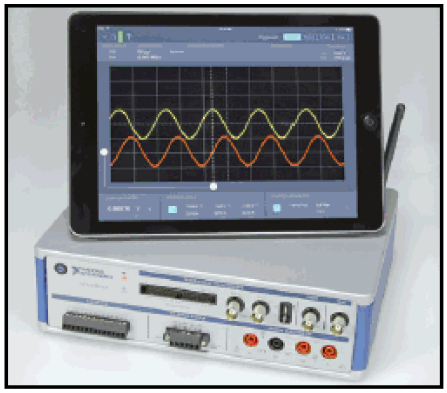




Projects







Review

Industry

DesignSpark

Labs

Regulars

Professional Lab Power Supply

Quality has its price

By **Arne Hinz** and **Martin Christoph**

Ordinary power supplies are as common as sand on the beach, and even lab power supplies come in all sorts and sizes. These are adequate for many purposes, but if you want high quality, high stability and precise regulation, you have to put out a lot of money—and that for a power supply with only modest output power. Another option is to build one yourself.

If you regularly put together electronics projects, a small power supply with an output current rating of around 1 A and an output voltage adjustment range of at least 1 V to 12 V or so can certainly come in handy. That should be enough for most of the small circuits you build yourself. Small adjustable power supplies of this sort with regulated output voltage and an enclosure are available from electronics mail-order companies at low prices, and most of them have displays

that show the output current and voltage at the same time.

However, for the rest of your projects you need something more. This is especially true for ambitious hobbyists who tackle relatively complex electronics projects, as well as professional development labs where good lab power supplies are an essential part of the basic test equipment and a lot of money is spent on professional equipment. The question of which power supply features are generally important or relevant is, as in so many cases, not so easy to answer because the requirements depend on the application. However, it's certainly worth having a close look at the vast numbers of laboratory supplies that offer a lot of power at relatively low cost.

For less than 100 dollars you can buy power supplies made in Asia with an adjustable voltage range from 0 V (or close to that) to 30 V and adjustable current limiting from (nearly) 0 A to around 3 A, and which also have digital displays, remote control and other features. For 50 dollars more, you can even get a switching power supply with nearly 600 watts of output power. However, the adage "cheaper is better" is far from true here. You don't have to be a genius to realize that at these prices, it's not possible to combine good basic specs with other important features such as good efficiency, good voltage regulation or durability. For power supplies, quality means complexity and complexity means cost.

Although the power supply described in this article won't beat units in the four-figure price range, it has very good specs.

Features

- Efficient lab power supply with switching preregulator
- Operates from the 12 V output of a PC power supply
- Efficiency up to 70%
- Full electrical isolation with multiple modules
- Output voltage 0–30 V, adjustable in 10 mV steps
- Output current 0–1 A, adjustable in 10 mA steps
- Voltage and current shown on LED displays
- Good load regulation
- Low drift
- Fast shutdown button

Basic design

Good laboratory power supplies have two common characteristics: they stay relatively cool and they deliver a stable output voltage. The first of these requires switching technology, since high power dissipation (heat inside the power supply) reduces the lifetime and degrades the output voltage drift. The second means that the output voltage remains virtually constant over the entire load range and temperature range (static stability). With regard to dynamic stability, a fast and clean control response is important to keep overshoots and undershoots due to load changes or other causes as small as possible. Low ripple voltage and essentially zero output noise are also basic requirements. However, keeping high-frequency output noise under control is not especially easy with switching regulators. Finally, we can also mention a factor that is more related to practical use than to quality: if a power supply can provide more than one output voltage, the outputs should be galvanically isolated so they can be connected together in any desired arrangement without asking for trouble.

From all this, we can conclude that low cost and high quality (low noise and good regulation) are mutually contradictory requirements. The logical solution to this dilemma is to combine linear and switching technologies in order to reap the benefits of both, which calls for a more complicated design.

This takes the form of a conventional linear regulator in the output stage, which allows the voltage or current to be adjusted quickly and precisely and ensures low noise. Ahead of this linear stage there is a switching regulator that provides the input voltage for the output stage, which is always a bit higher than the output voltage but does not have the same high quality. The power dissipation of the linear output stage is low, even with low output voltages and high output cur-

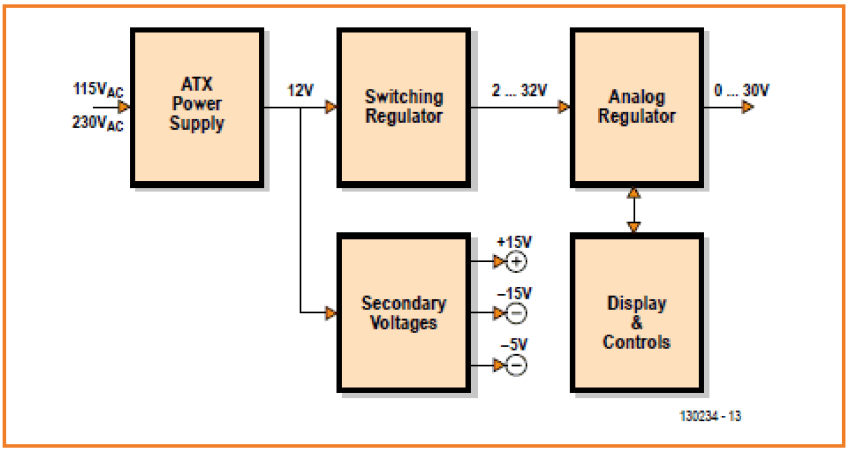


Figure 1.
As you can see from the block diagram, this is not a simple circuit.

rents, because the voltage drop over this stage is low. This requires more components than a simple design, so it is more complicated and more costly. Ready-made lab power supplies with this configuration are therefore distinctly expensive. The authors took this into consideration and gave careful thought to the design. Development started at the Institute for Rectifier Technology and Electrical Drives (ISEA) of the Aachen University of Applied Sciences (RWTH Aachen, Germany), and the design was subsequently refined by Arne Hinz in the electronics lab as a project for the practical part of his studies.

The block diagram in **Figure 1** shows the result. To make things easier for DIY construction, the switching regulator input stage is not designed for

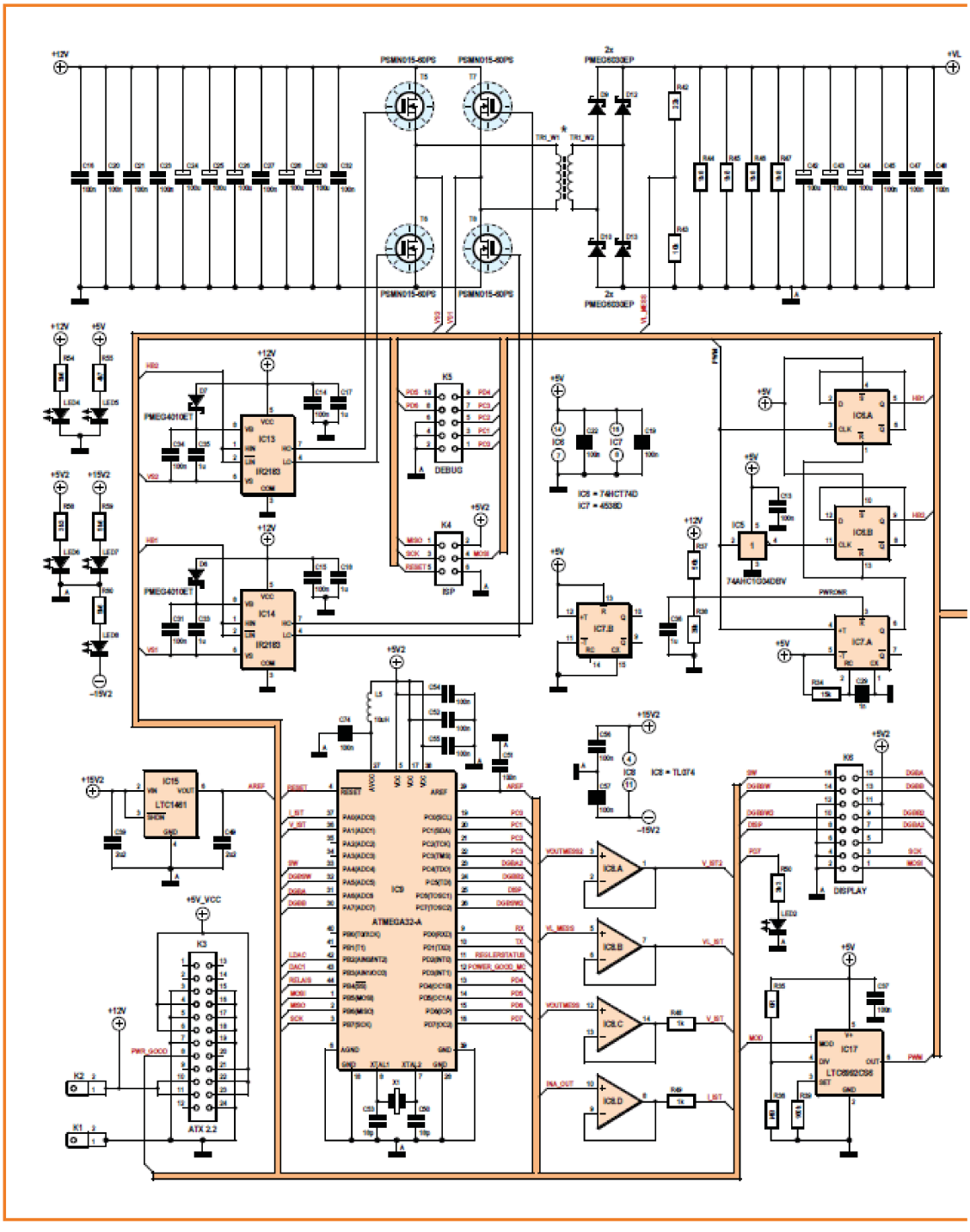
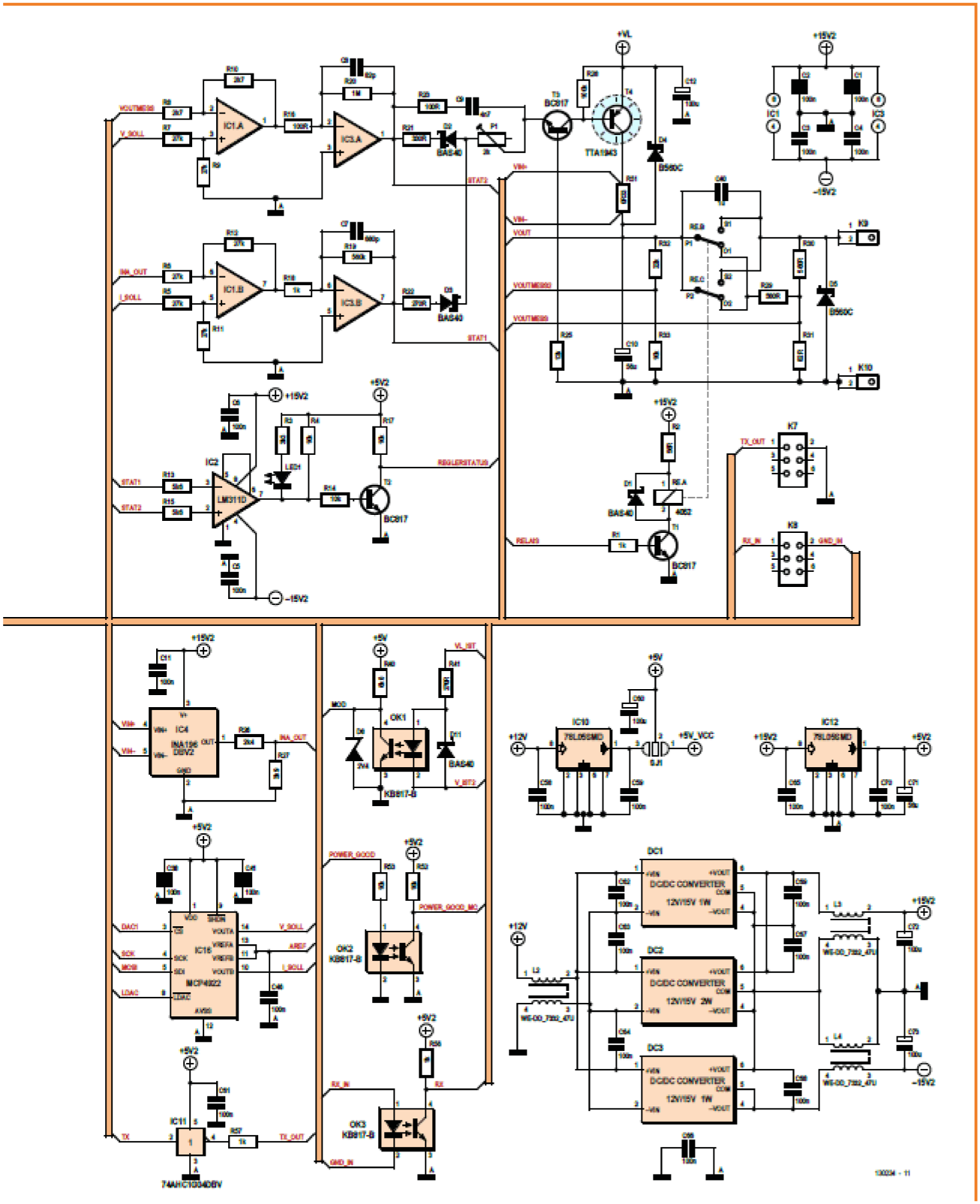


Figure 2.
The schematic of the lab power supply is fairly extensive, even without the display & control section.

direct off-line operation from AC power. Instead, it is designed for low-voltage operation with an input voltage of 12 V DC. This can be provided with sufficient quality and power at low cost by a simple PC power supply, and PC power supplies with capacities from less than 200 W to 1 kW are readily available. Strictly speaking, this means we have a three-stage power supply design. In the top row of the block diagram, from left to right, the first stage is a standard ATX power supply. It feeds the switching regulator in the middle, which acts as a sort of preregulator for the linear output stage to the right. At the bottom middle you see the block for the secondary voltages (±15 V and +5 V on-board supply voltages), and at the bottom right the display & control module with



the controls, indicators and displays for voltage and current. The advantage of this three-stage modular design is that you can obtain several galvanically isolated supply voltages by connecting several regulator modules (with combined switching and linear regulators) to a single ATX power supply in the same enclosure. Galvanic isolation is provided by the power transformer in the switching regulator stage in the middle. As a result, all output voltages can be connected together in any desired arrangement.

Detailed circuit description

First a recap: the high-performance lab power supply module consists of a switching regulator input stage with galvanic isolation and a linear regulator output stage. This arrangement is fed by a high-power 12-V source, such as a low-cost ATX power supply. The two regulators and the microcontroller that controls everything are assembled on a single PCB, with the display & control components on a second PCB. The result is a compact module, and several of these modules can be packaged in a conventional 19" rack-mount enclosure and powered collectively by a PC power supply.

But first let's look at the requirements, since quality is not simply a matter of using a 19-inch package or delivering a lot of output power. The regulation characteristics are what really matter. To keep the cost of this circuit design within reasonable bounds despite the relatively high complexity, the output power is dimensioned to be sufficient for many common lab applications: maximum output voltage 30 V, maximum output current 1 A. With a maximum output power of 30 W, a relatively small transformer can be used in the switching regulator stage. What's more, we have designed a transformer that is very easy to make yourself—more about that later.

12 V input

The schematic diagram in **Figure 2** is a bit daunting at first glance. However, it's not as bad as it looks, even if there isn't much white space on the drawing. Connector K3 at the bottom left is a standard ATX 2.2 power supply connector, as found on every PC motherboard. The main power connector of an ATX power supply can therefore be plugged in directly. K3 is wired so that the PC power supply delivers output power when it is plugged in to K3 and switched on (AC power present). This means that the AC power switch of the PC power supply is the AC power switch for the entire lab power supply. The lab power supply is supplied from this 12-V rail. The two blade connectors K1 and K2 can be used to daisy-chain the 12 V rail to other lab power supply modules.

Secondary voltages

The circuitry for this block in Figure 1 is located in the bottom right corner of the schematic diagram in Figure 2. Here the DC/DC voltage converters DC1, DC2 and DC3 generate the ±15 V supply voltages for the opamps and other components. DC1 and DC2 are wired in parallel for the +15V2 (15.2 V) supply rail; DC3 handles the –15V2 supply rail on its own. The converter modules also provide galvanic isolation. These are switching converters, so their noise is suppressed by choke L2 at the input and chokes L3 and L4 at the output. Otherwise the high-frequency noise would penetrate to the output voltage of the power supply. For proper understanding of the schematic diagram, it is essential to know that two different ground symbols are used here. The usual ground symbol relates to the 12 V input side, while the symbol with the "A" superscript relates to the galvanically isolated output side. To avoid noise problems, high-voltage capacitor C40 provides high-frequency coupling between the two ground levels.

There are two on-board 5 V supplies. Most of the digital circuitry, including the microcontroller (IC9) and the display & control section (in Figure 3), is powered from +5V2, which is derived from +15V2 by a small linear regulator (IC12). A separate +5 V supply voltage referenced to the input ground is also necessary on the input side. It can be taken directly from the 5 V output of the PC power supply or derived from its 12 V output by another voltage regulator (IC10). Since some PC power supplies have poor 5 V regulation when the 5 V output is lightly loaded and the load on the 12 V output is variable, it is preferable to connect pins 2 and 3 through the solder bridge SJ1.

Switching regulator

The circuitry for this block in Figure 1 is spread out over the schematic diagram in Figure 2. First let's look at the voltage conversion circuitry. The 12 V input section, with a massive array of decoupling capacitors, is located at the top left of the diagram. The bank of five relatively small electrolytic capacitors (C24–C26, C28 and C30) and six multilayer capacitors wired in parallel drastically reduces the high-frequency input impedance, pro-

viding very good decoupling of the current spikes coming from the PC power supply. The same strategy is used on the secondary side, since parallel arrangements of this sort have better dynamic characteristics than a single capacitor with the same total capacitance.

The four power MOSFETs form a bridge circuit driven by the switching regulator IC LTC6992 (IC17). A pair of IRF2183 MOSFET drivers (IC13 and IC14) are located between the voltage-controlled PWM output of the regulator IC and the gates of the MOSFETs. They provide sufficient drive current and have built-in dead time to ensure that both MOSFETS of a half-bridge section cannot conduct at the same time. Each driver IC has a bootstrap circuit consisting of a Schottky diode (D6 or D7) and storage capacitors (C31/C33 or C34/C35) to provide sufficient drive voltage for the gates of the high-side MOSFETs. IC17 only has a single 300 kHz PWM output, so a clock signal with opposite phase is generated by inverter IC5. The two D-type flip-flops (IC6) supply control signals at 150 kHz with an offset depending on the duty cycle.

Now comes the question of how the switching regulator adjusts its output voltage so that it is always slightly higher than the output voltage of the linear regulator. For this purpose, the control signal MOD (which comes from the optocoupler OK1) is generated for IC17. The emitter LED of the optocoupler is simply driven by the current through series resistor R41 resulting from the difference between the output voltage of the switching regulator and the output voltage of the linear regulator. The MOD signal controls the output voltage of the switching regulator to maintain a voltage difference of approximately 1.5 to 2 V. The VL_MESS signal from the switching regulator output is taken from voltage divider R42/R43, while the VOUTMESS2 signal from the linear regulator output is taken from voltage divider R32/R33. The two signals are buffered by opamps IC8a and IC8b and applied to optocoupler OK1 as VL_IST and V_IST2.

Linear regulator

The output stage of the linear regulator is built around the PNP power transistor T4, which is wired in common-emitter configuration and

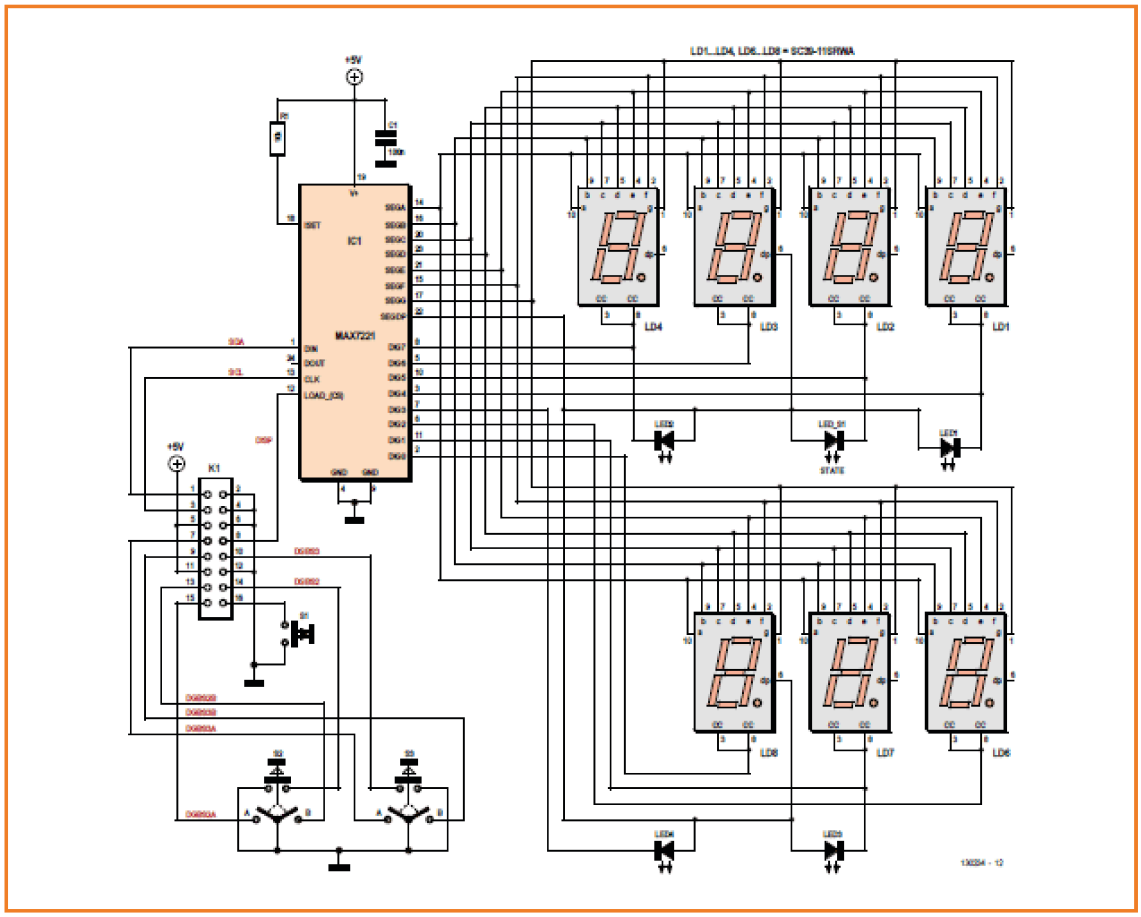


Figure 3.
Schematic of the display & control section with LED displays, LEDs and rotary encoders is significantly simpler.

driven by transistor T3. The emitter current of T3, which is the same as the base current of T4, is determined by the voltage and current regulators. It flows through P1, which is adjusted so that T4 can only become slightly saturated. This improves the regulation characteristics. The two diodes D2 and D3 in combination with resistors R21 and R22 form an override circuit for current regulation. This works as follows: As long as the output current is below the set limit, the voltage regulator (IC1a and IC3a) controls the base current of T4 and thereby the output voltage, since the voltage at the output of the current regulator (IC1b and IC3b) is "logic low" and D3 is reverse biased. However, if the output current becomes too high due to a low-resistance load, the output of the current regulator overrides the output of the voltage regulator, raising the voltage of the drive signal. As a result, the output voltage drops to the level necessary to keep the current at the limit value. Current regulation takes precedence because R22 has a lower resistance than R21.

In the voltage regulator, IC1a acts as a differential amplifier. Its inputs are the VOUTMESS signal from the output voltage (via R29, R30 and R31) and the setpoint value V_SOLL, which

Component List: Supply

Resistors
(default: 0.1W, 1%, SMD 0603)

R1,R18,R48,R49,R56,R57 = 1kΩ
R2 = 56Ω 5%, 0.33W, SMD 0805
R3,R50,R58 = 3.3kΩ
R4,R14,R17,R33,R43,R52,R53 = 10kΩ
R5,R6,R7,R9,R11,R12 = 27kΩ 0.1%
R8,R10 = 2.7kΩ, 0.1%
R13,R15,R54,R59,R60 = 5.6kΩ
R16,R23 = 100Ω
R19 = 560kΩ
R20 = 1MΩ
R21 = 330Ω
R22,R41 = 270Ω
R25 = 12kΩ
R26 = 2.4kΩ
R27 = 3.9kΩ
R28 = 100kΩ
R29,R30 = 560Ω, 1.5W, SMD 2512
R31 = 82Ω, 0.25 W, SMD 1206
R32,R42 = 22kΩ
R34 = 15kΩ
R35 = 0Ω (wire link)
R36 = not fitted
R37 = 56kΩ
R38 = 39kΩ
R39 = 160kΩ
R40 = 6.8kΩ
R44–R47 = 1.8kΩ 1W, 5%, leaded
R51 = 0.33Ω 2W, SMD 2512
R55 = 4.7kΩ
P1 = 2kΩ, multiturn trimpot, tall (9353755)

Capacitors
(default: 50V, X7R, SMD0603)
C1–C6,C11,C13,C14,C15,C19,C22,C31,C34, C37,C38,C41,C46,C51,C52,C54–C59,C61–C70,C74 = 100nF
C7 = 680pF, NP0
C8 = 82pF, NP0
C9 = 4.7nF
C10,C71 = 56µF 63V, 3.5mm pitch, 8mm diam.
C12,C24,C25,C26,C28,C30,C42,C43,C44,C60,C72,C73 = 100µF 63V, low ESR, 3.5mm pitch, 8mm diam.
C17,C18,C33,C35,C36 = 1µF, X5R
C16,C20,C21,C23,C27,C32,C45,C47,C48 = 100nF, SMD 1206
C29 = 1nF
C39,C49 = 2.2µF 50V, X5R, SMD 0805
C40 = 1µF 100V, PET, 7.5mm pitch
C50,C53 = 18pF
C66 = 100nF 630V, SMD 1812

Inductor
L2,L3,L4 = 47µF 0.9A, dual choke (1869658)
L5 = 10µF, SMD 0805 (Reichelt # JCI 2012 10µ)

Semiconductors
D1,D2,D3,D11 = BAS40W, SMD SOT323 (8734380)
D4,D5 = B560C, Schottky, SMD SMC (1858602)
D6,D7 = PMEG4010ET, Schottky (2311223)
D9,D10,D12,D13 = PMEG6030EP, Schottky, SMD SOD128 (1829207)
D8 = BZV55-C2V4, zener diode, SMD Mini-MELF (1097193)
LED1, LED2, LED4–LED8 = LED, orange, SMD 0603
OK1,OK2,OK3 = KB817-B, DMD4, optocoupler
T1,T2,T3 = BC817, SMD SOT23
T4 = TTA1943, PNP, TO-3P (1901958); alternative: 2SA1943
T5–T8 = PSMN015-60PS, n-channel-MOSFET, TO220 (1845643)
IC1,IC3 = TL5580, SMD SOIC8 (1755396)
IC2 = LM311D, SMD SOIC8 (2293183)
IC4 = INA196AIDBVT, SMD SOT23-5 (1564942)
IC5,IC11 = 74HC1G04GW, SMD SOT353 (1085251)
IC6 = 74HCT74D, SMD SOIC14 (1085304)
IC7 = 74HCT4538D, SMD SOIC16 (1631658)
IC8 = TL074, SMD SOIC14 (1459705)
IC9 = ATmega32-A, SMD 44TQPF, programmed, Elektor Store # 130234-41
IC10,IC12 = 78L05, SMD SOIC8
IC13,IC14 = IR2183SPBF, SMD SOIC8 (1023247)
IC15 = LT1461CCS8-4, 4,096 V, SMD SOIC8 (1663430)
IC16 = MCP4922-E/SL, SMD SOIC14 (1332114)
IC17 = LTC6992CS6-2, SMD TSOT23-6 (1848046)
DC1,DC3 = TMA 1215S, 12V/15V, 1W, Traco (1007521)
DC2 = TMH 1215S, 12V/15V, 2W, Traco (1007560)

Miscellaneous
X1 = 16MHz quartz crystal, SMD HC49
Heatsink for T4, Fischer SK 08, 3.2K/W (Reichelt V 4511D)
Heatsink for T5–T8, Fischer SK 125 84, 5.8 K/W (4621335)
K5 = 10-pin boxheader
K6 = 16-pin boxheader
K4,K7,K8 = 6-pin boxheader
K3 = 24-pin ATX-2.2 plug, PCB mount (2113352)
K1,K2,K9,K10 = Spade terminal, PCB mount, 4.8 x 0.5mm (4215552)
RE = relay, 12V, 2 x c/o (Reichelt FIN 40.52.9 12V)
2 pcs ferrite core E32/6/20-3F4 (3056107)
Cable ties for securing cores
Washer, ceramic, TO-3P, for T4 (RS Components 283-3830)
Washer, ceramic, TO220, for T5–T8 (RS Components 177-7767)
Screws and nuts for heatsinks and T4–T8
19-inch case, 3 rack units (3U) with:
Module carrier 3U, e.g. Hammond 84TE 235mm Typ F
2 pcs support rail, length 220mm
PCB #130234-1 and 130234-3

7-digit numbers in round brackets are Newark/Farnell order codes

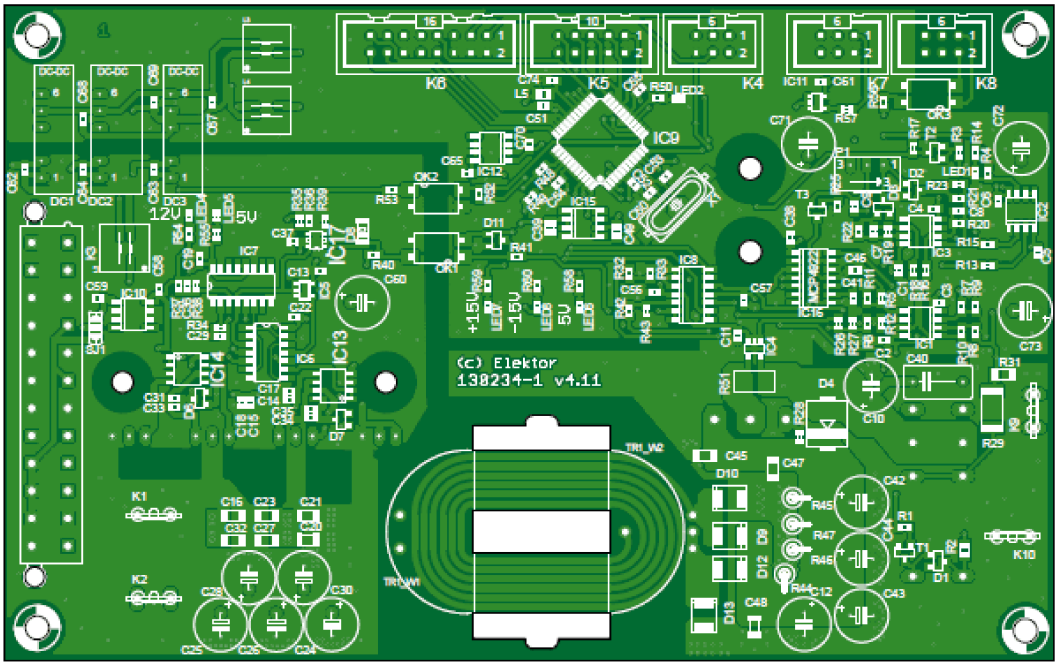


Figure 4.
Component layout of the regulator board for the circuit shown in Figure 2. The power transistors are mounted on the rear.

comes from the dual DAC IC16. In the same way, IC19 provides the current setpoint value I_SOLL to IC1b, where it is compared with the signal INA_OUT. The latter signal represents the output current and is obtained by converting the voltage drop over the low-resistance sense resistor R51 into a ground-referenced voltage. This is handled by the INA196 (IC4), which is specifically designed for this purpose.
Of course, the DAC has to get data from some-

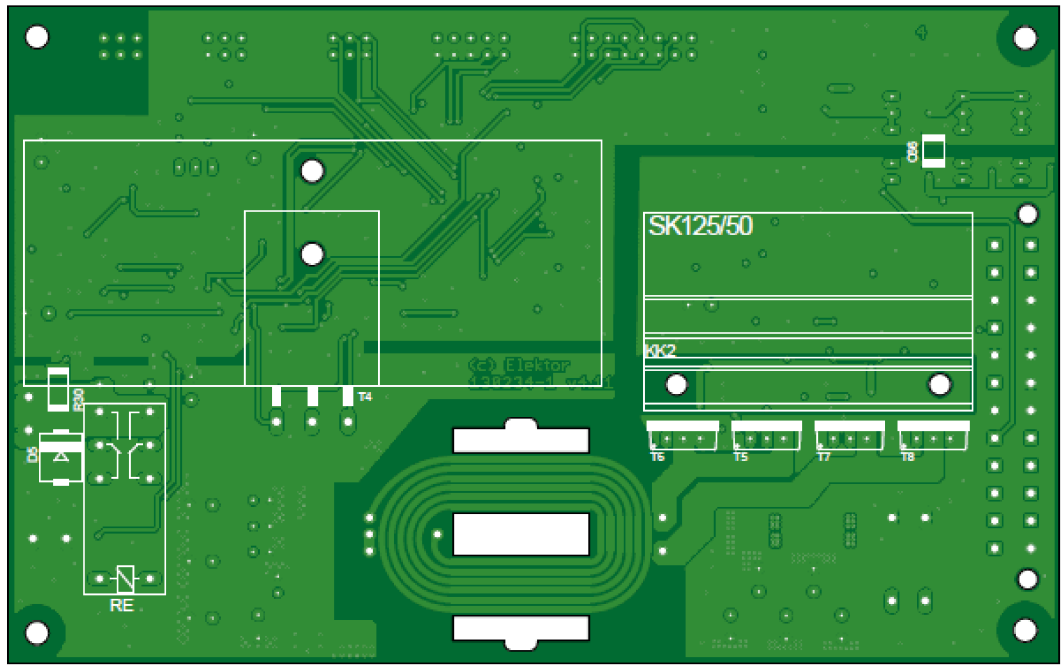


Figure 5.
Rear side of the regulator board. Here you can see how the heatsinks and board are screwed together and where the power transistors are located.

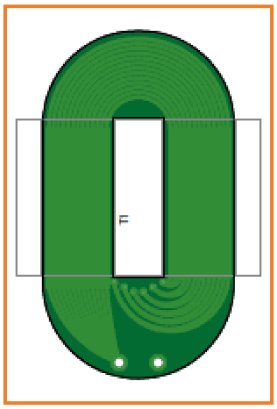


Figure 6.
The secondary winding PCB for the planar transformer. Like the regulator board, it has four layers.

where so it can provide the setpoint values. This is where the microcontroller (IC9) comes into play. It receives voltage and current signals which are buffered by opamps IC8c and IC8d and applied to two internal inputs as V_IST and I_IST. IC9 measures the voltage levels of these signals and outputs the measured values to the display. The microcontroller also outputs the setpoint data corresponding to the user settings to the DAC IC16. Above K3 you can see IC15 (LT1461), which is a precision voltage regulator that supplies a reference voltage of 4.096 V with a tolerance of 0.08% and a drift of only 12 ppm/°C (with the specified version). It is not essential to use this high-grade version of the IC, since the overall accuracy depends more on the precision of the resistances of the voltage dividers used to measure the voltage (R30/R31 and R32/R33) and the current (R26/R27 and R51).

Display & control

The circuit diagram of the display & control section in **Figure 3** includes the displays, drivers, LEDs and controls. Connector K1 is connected to K6 of the main circuit board (Figure 2). The MAX7221 driver IC receives data from the microcontroller in Figure 2 over a serial link. It controls a four-digit seven-segment LED display for the voltage and a three-digit display for the current, using multiplexed signals. For both voltage and current the decimal point is hard-wired with two decimal places. Since this means that the driver does not have to handle any decimal points, the corresponding outputs are used to drive the four LEDs (LED1–LED4) and the LEDs integrated into S1. The red LEDs on the right side are lit alternatively when the voltage regulator is active (LED2) or the current regulator is active (LED4). The green LEDs on the left side light up when a new current or voltage is setting is initiated by pressing S3 or S2. Button S1 lights up when the output voltage is present.

Miscellaneous

A relay is necessary to allow the output voltage to be switched on and off quickly with S1 (Figure 3). In Figure 2 it is driven by T1, which in turn is driven by a digital output of the microcontroller via R1. LEDs LED4–LED8 indicate the presence of the input voltage and the secondary voltages. Diodes D4 and D5 protect the circuitry against reverse currents due to incorrect connection of the output terminals, which can easily happen in lab situations (e.g. when working with batteries). The response times of the regulators are tuned by the RC networks R20/C8 (voltage) and R19/C7 (current). The comparator circuit built around IC2 checks whether voltage regulation or current

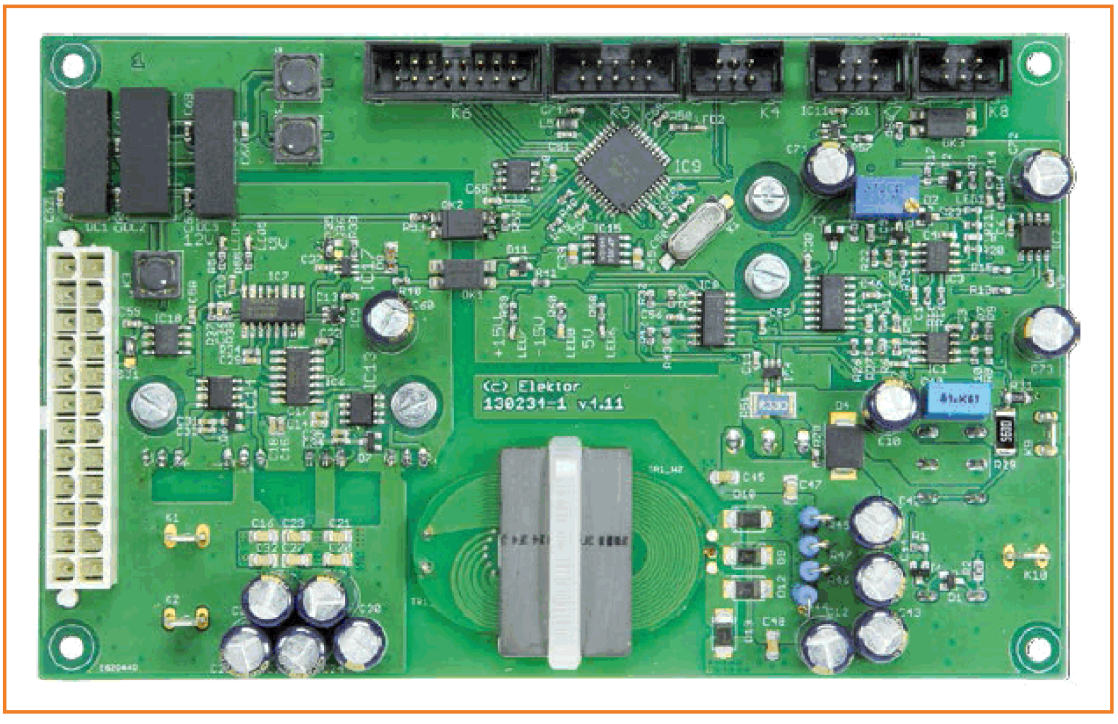


Figure 7.
Photo of the fully assembled regulator board. This gives you an impression of the planar transformer.

regulation is active and informs the microcontroller via T2. LED1 is lit when current regulation is active. A switch-on delay is implemented using the monostable IC7a and the reset pin of IC6, so that the switching regulator does not start up until all of the supply voltages have stabilized.
Connectors K7 and K8 can be used for serial data communication with the USART of the microcontroller. Among other things, this could be used to link several power supply modules in order to generate balanced supply voltages. The input through K8 is galvanically isolated by OK3, enabling serial linking without galvanic connection of the modules. However, this interface is not presently integrated into the firmware, so if you want to use it you will have to do a bit of programming. Several I/O pins of the microcontroller are brought out to K5. They can be used for any desired purpose by extending the firmware. The microcontroller can be programmed over K4 with the usual in-system programming devices. Transformer TR1 is a planar transformer. Its construction is described below. The turns ratio of TR1 is 2:3.

Construction

There are individual PCBs for the circuits in Figure 2 and Figure 3. The massive use of SMD components in 0603 format is a sure sign that you will need a lot of soldering experience, among other things. A project of this complexity is definitely not a good choice for beginners, and even then you should not tackle it without assistance. For experienced builders, it's surely unnecessary to say that you should fit the low-profile components first and then the higher components.
With the regulator board shown in **Figure 4**, there are three things that require special attention. The first two are apparent when you look at the rear of the board (**Figure 5**): transistors T4 and T5–T8 are mounted on the rear side. It is also necessary to fit insulating spacers on the screws connecting the heat sink for T4 to the PCB, to prevent it from causing short circuits on the board. Things are easier with the heat sink for the MOSFETs because it can be fastened directly to the PCB with short M3 screws. The MOSFETs can be mounted with screws or with suitable mounting clips, which is easier. All five transistors must be insulated from their heat sinks by thermal pads.
The most unusual thing about this project is unquestionably the planar transformer. It is built with a pair of ferrite pot cores, and the clever part is that you don't have to wind anything because the primary winding is integrated in the PCB. There is a separate small PCB for the secondary winding (**Figure 6**), which is connected to the main board by thick copper wires and glued

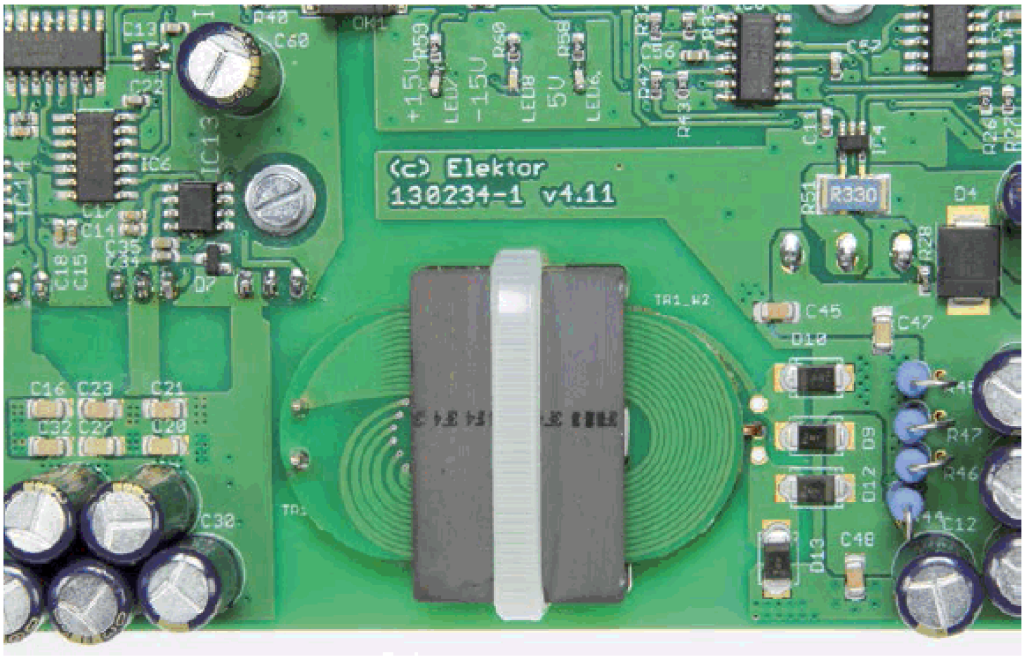


Figure 8.
Close-up view of the planar transformer. Here you can see how the small PCB with the secondary winding is fitted.

onto the main board with a layer of Kapton tape. The tape must be kept out of the mating areas of the pot cores. Although this "printed" transformer does not utilize the full winding volume of the cores, it saves a lot of manual labor. To ensure that this works properly, both PCBs are built with four layers. Making your own PCBs for this project is therefore not recommended. The four layers provide four parallel windings, which reduces the negative impact of the skin effect at high frequencies. After fitting the small PCB with the secondary winding, position the two pot core sections so they pass through the holes in the regulator board and secure this assembly with a cable tie. The picture of the prototype in **Figure 7** and the close-up photo in **Figure 8** show what it should look like.

The PCB for the display & control section (**Figure 9**) has a significantly simpler layout and is built as a conventional two-layer board. There are only a few special aspects here. One is that K1 is mounted on the rear of the PCB, and another is the reason for the two round holes in the board. Maybe you guessed already: they are cutouts for 4-mm jacks mounted on the front panel of the enclosure. These jacks should be connected to K9 and K10 on the regulator board by lengths of stranded wire with insulating sleeves. **Figure 10** shows the finished combination of regulator board and display & control board.

A 19-inch rack-mount enclosure with 3-unit height is a good choice for housing a decent lab power supply. It provides enough space for the PC power supply as well as several lab power supply modules. The PCBs have been specifically designed for this.

Adjustment and operation

After the boards are assembled, checked and connected by a ribbon cable, all that's missing is the microcontroller firmware. It was written in C with AVR Studio and can be downloaded from the Elektor web page for this article, along with the layout files for the PCBs. The microcontroller can be programmed using an ISP device, such as the Atmel AVRISP MkII or similar. Of course, the microcontroller must be powered up before it can be programmed, which means that the board must be connected to a 12 V supply.

After the microcontroller has been programmed, you should see something reasonable on the dis-

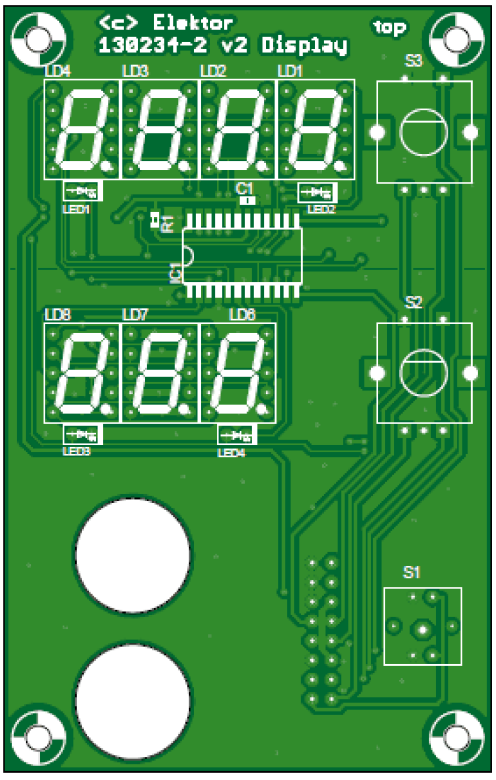


Figure 9.
Component layout of the display & control PCB. The board is a conventional two-layer design.

Component List: Display

Resistors
R1 = 10kΩ 0.1W, 1%, SMD 0603

Capacitors
C1, = 100nF 50V, X7R, SMD 0603

Semiconductors
LD1–LD4,LD6,LD7,LD8 = SC39-11GWA, LED-Display, red , common cathode (2314233)
LED1,LED3 = LED, green, rectangular, leaded (1142607)
LED2,LED4 = LED, red, rectangular, leaded (1581150)
IC1 = MAX7221CWG+, SMD SOIC (9725725)

Miscellaneous
S1 = pushbutton with LED, Multicomp MC-SPHN3-YCA043T (2146950)
S2,S3 = rotary encoder with switch, (Mouser 652-PEC12R-4220F-S24)
Knobs for rotary encoder
Caps for knobs
K1 = 16-pin boxheader
2 pcs 16-way IDC socket for flatcable
16-way flatcable
PCB # 130234-2

7-digit numbers in round brackets are Newark/Farnell order codes

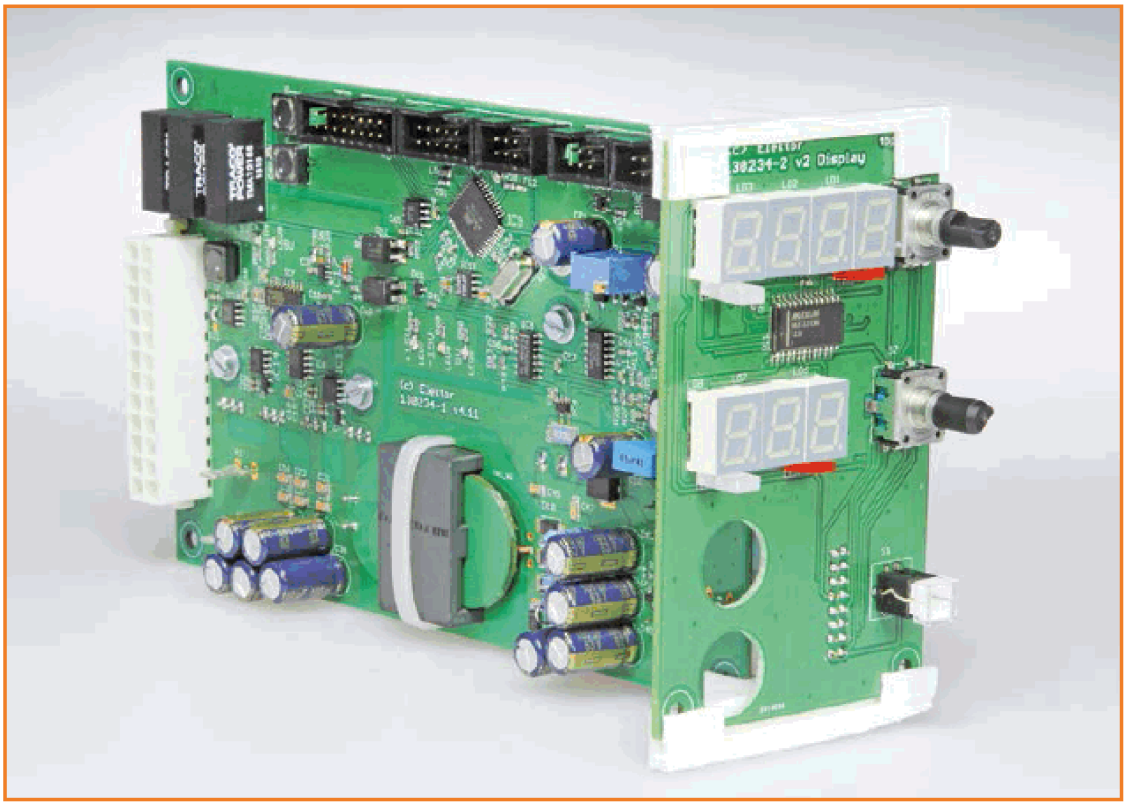


Figure 10.
The finished prototype, consisting of the regulator board and the display & control board.

play. When you press rotary encoder S3 or S2, the corresponding green LED lights up to indicate that you can set the voltage or current by turning the encoder knob. Each encoder increment corresponds to 10 mV or 10 mA. If you turn encoder S3 while it is pressed, the voltage increment is 1 V, and when S2 is turned while pressed the current increment is 100 mA. Pressing S3 or S2 again exits setting mode; the setting is accepted and the green LED goes dark. After this the actual voltage and current values are displayed again. To access the calibration mode, press S1 and turn S2 at the same time. Now you have to apply a load to the output of the power supply by connecting an ammeter, which effectively short-circuits the output. A meter with a measuring range of 2 A is ideal. In this mode the output voltage is set to 1 V and the current limit is set to maximum. The display shows the measured current and a correction factor instead of the voltage. LED1 indicates the sign of the correction factor. First turn P1 about one to two turns past the point where the current stops rising. Then you can use S3 to change the compensation factor in steps of 0.1 percent. Adjust the compensation factor so the displayed current matches the current read from the external ammeter. Exit this mode by again pressing S1 and turning S2. Current measurement is affected by component tolerances and the minimum load current through R29–R33. **Figure 11** gives an impression of the regulation characteristics of the lab power supply for a load change from 90% to 10% at a voltage of 15 V.

(130234-I)

Web Link

www.elektor-magazine.com/130234

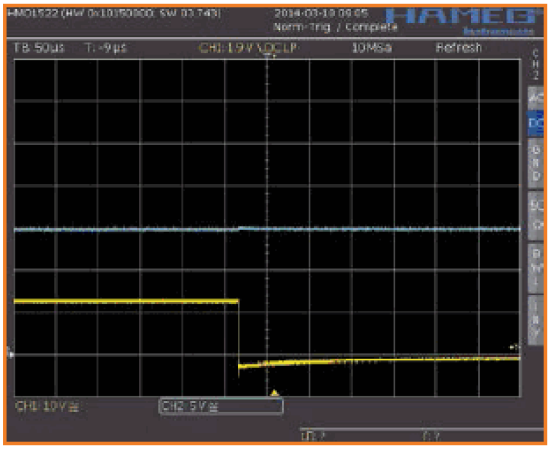
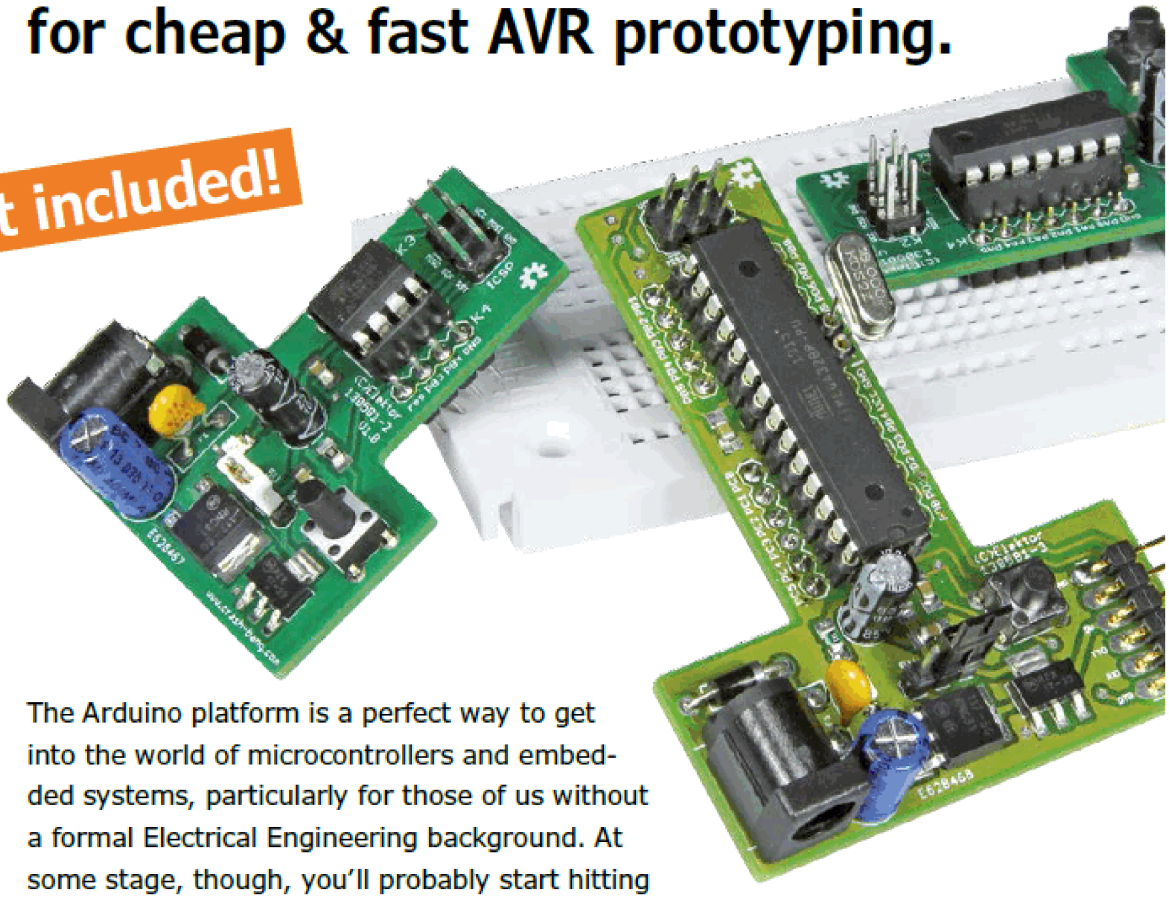


Figure 11.
As you can see from this oscilloscope screenshot, the output voltage (15 V) remains perfectly stable when the load is suddenly reduced from 90% to 10%.

T-Boards 8/14/28

Three sizes for cheap & fast AVR prototyping.

T-shirt included!



By **Andrew Retallack**
(South Africa)

The Arduino platform is a perfect way to get into the world of microcontrollers and embedded systems, particularly for those of us without a formal Electrical Engineering background. At some stage, though, you'll probably start hitting constraints and want make the leap to working with 'raw' microcontrollers. The T-Board makes that jump just a little less daunting.

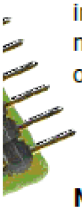
One wouldn't want to belittle the design of the Arduino boards—without the Arduino the Maker movement would be a fraction of its size and far less people particularly youngsters would be comfortable using microcontrollers. However, as your projects become more ambitious, at some point you may need to make the move away from the Arduino platform. More complex projects, particularly those that need a specific physical form, are often best implemented with custom PCB designs that directly incorporate the microcontrollers. Some projects need to meet specific constraints, which require flexibility the Arduino can't provide—for example, more power-efficient designs by operating at lower voltages or at slower clock speeds. For less complex projects a smaller Atmel AVR microcontroller is better suited to the task and more cost effective than a full Arduino. Whatever the reason, the first step most of us would take in a prototyping process would be to set the microcontroller up on a breadboard.

Microcontrollers on breadboards: a challenge

On the face of it, it doesn't take much to build your own microcontroller project on a breadboard—there are plenty of online resources available [1], and the supporting components are inexpensive. However, once you actually start

working on a project, it becomes clear that there are a number of challenges. For starters, a basic microcontroller setup already has a number of jumper wires, capacitors, resistors and connectors infringing on your limited breadboard workspace. This not only limits the number of available connections, but makes it harder to trace your connections and carry out any troubleshooting.

Secondly, all the pins on the microcontroller are accessible (which is something you don't always want) and are unlabeled. Consequently extra care and a lot of pin-counting to connect components up correctly and to keep the white smoke from escaping from inside your MCU! Finally, and importantly, is the programming—whether you use an FTDI or ISP programmer, you're adding even more to the breadboard just to get the microcontroller working. The result is a rat's nest of wires and components with temperamental connections and an experience that isn't as fun and rewarding as you'd like!



Making prototyping simpler: the T-Board

It was from the author's experiences and frustrations with breadboard prototyping that the T-Board design emerged: a breakout board that would speed up microcontroller prototyping by reducing the complexity whilst retaining flexibility. Three versions of the board were designed, to accommodate Atmel's more popular AVR microcontrollers: the 28-pin ATmega range (ATmega8/48/88/168/328), the 14-pin ATtiny range (ATtiny20/24/44/84/441/841) and 8-pin ATtiny range (ATtiny13/25/45/85). All three versions are breadboard-friendly, can be self-powered at 3.3 V or 5 V, and contain ICSP headers for easy programming.

The electrical design

T-Boards come in three flavors: '8', '14' and '28'; you select the one that best suits your requirements. Because there are three boards, there are also three schematics: **Figures 1** (T-Board 8), **2** (T-Board 14), and **3** (T-Board 28).

Apart from the obvious differences due to the microcontrollers used, the schematics are largely identical. Power is supplied through a standard 2.1-mm center-positive jack K1, with diode protection (D1), a PTC resettable fuse with a 1-A trip rating (F1), and filtering capacitor (C1). The two voltage regulators IC1 and IC2 are low dropout regulators with fixed 5-V (NCP1117DT50G) and 3.3-V (LD1117S33TR) outputs, with maximum input voltage of 20 V and rated output in excess of 1 A. The jumper JP1 selects the output voltage. The T-Board 28 with its AVR '328 can also be powered from the FTDI board, assuming a 5 V supply.

The microcontrollers are mounted in DIL sockets, to allow them to be exchanged—there are a number of MCUs with compatible pin configurations. A 0.1-µF decoupling capacitor (C4) is placed close to the microcontroller's VCC and GND pins. The ICSP header connection is a standard 6-pin one, and powers the board during programming. Be

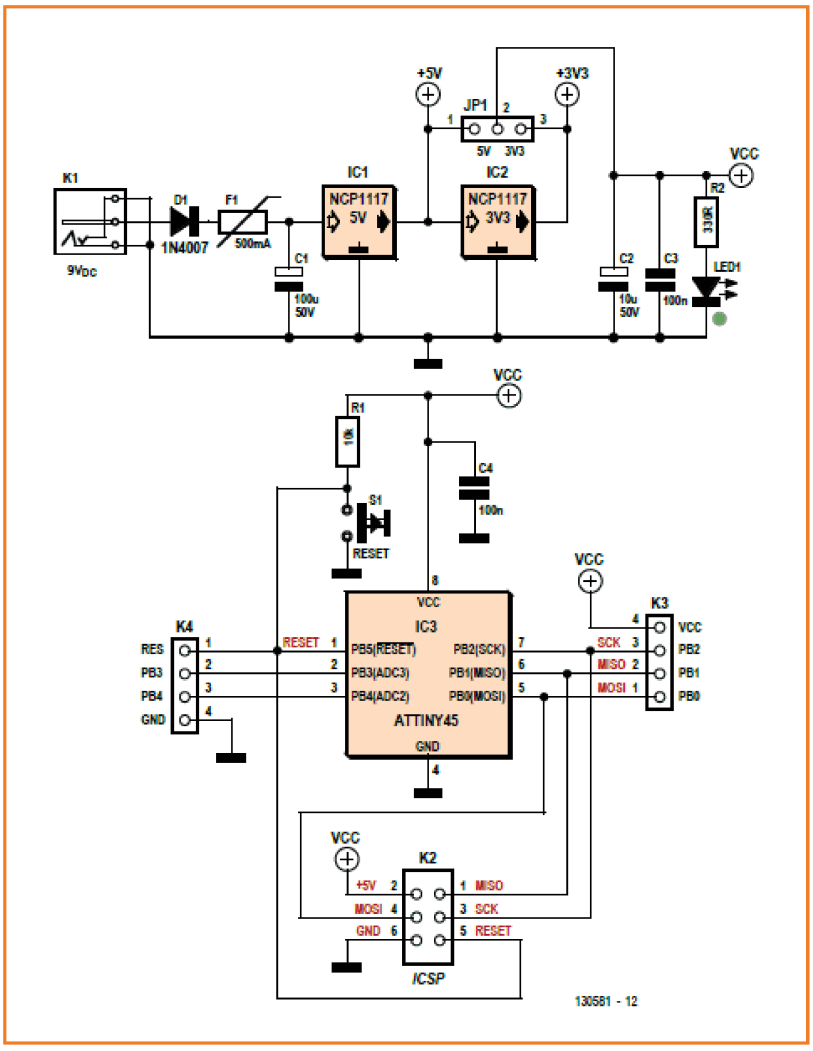


Figure 1.
T-Board 8 schematic. Tiny indeed—that '45 with just 8 pins.

aware that the voltage will be determined by the ISP programmer. A reset button, S1, that pulls the Reset pin low completes the design elements common across all three boards.

The real boards

The physical design of the T-Boards is relatively straightforward, as can be seen from the photographs in **Figures 4**, **5**, and **6**. Shaping was the key challenge, and the author went through various iterations in an attempt to achieve a balance between flexibility, simplicity, footprint size and future-proofing. The T-shape of the board was arrived at to keep as much as possible off the breadboard in order to free up working area—the power management and FTDI header (in the case of the T-Board-28) all sit off the breadboard. The footprint could have been made reduced further (currently two rows of the breadboard are occluded on each side of the center channel), but that would have been trade-offs. Flexibility could have been sacrificed by using a smaller surface-mount MCU, losing the ability to interchange different PDIP controllers. Alternatively, the smaller footprint would have been at the cost of simplicity, by moving the microcontroller off

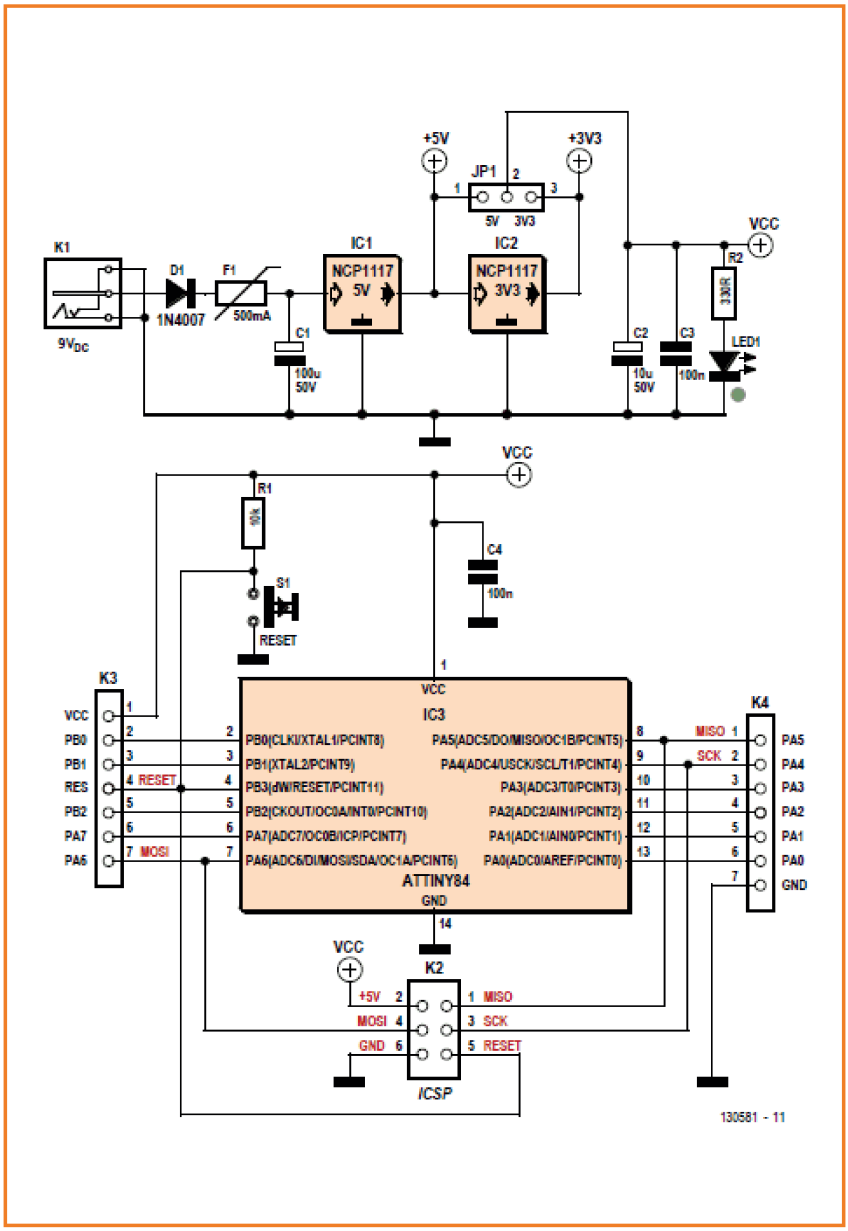


Figure 2.
T-Board 14 schematic. The mid-size option, based on the ATmega48 micro.

Choosing your IDE

Choosing an IDE is a very personal decision, and to recommend one outright is guaranteed to upset users loyal to Windows *vs* Mac *vs* Linux, or open a bouncy debate about the merits of open source *vs* commercial software, and bloated *vs* lightweight GUIs. In the end, most of the options available will be fine for those starting out with MCU development. It's when you get into serious embedded systems that require high levels of speed/memory/power optimization that more in-depth investigation is needed. To get you started, here are a few options:

Arduino IDE

Yes, you can use the Arduino IDE [3] to work directly with the raw MCU. You will be limited to those MCUs that the Arduino IDE supports, but you can find online resources to support a number of additional MCUs such as the ATtiny range.

The pros: working in a familiar environment; good library support; simple to configure; open source; Windows/MacOS/Linux support.

The cons: limited flexibility; limited MCU support; lack of advanced features; no debugging.

Eclipse with AVR Plugin

Eclipse [4] is a very popular open source IDE, supports many programming languages and runs on Windows, MacOS and Linux.

The flexibility and wide range of applications comes at a cost – it requires a little time to get installed and configured, and some understanding of the toolchain (compiler,

the board (either as a vertical plug-in module, or a move to a 4-layer board). In the end, for what the author was looking to achieve, we believe the design is a good balance.

In selecting components, a combination of through-hole and larger 1206 SMD components were chosen to allow for easy hand soldering. Certain readers may want to assemble their own though without needing to get their hands on reflow ovens.

T-Boards are available ready-assembled from the Elektor Store at prices that should defy home assembly from parts. For completeness' sake, in good Elektor tradition, and to serve all e-die-hards insisting on home-construction-all-the-way the three component overlays and associated parts lists are printed jointly in **Figure 7**. Also, the PCB artwork can be downloaded from the Elektor Magazine website [10]. Want to build it yourself? Go ahead.

The T-Board 28

The T-Board 28 with its ATMega328 micro has two features not present on the T-Boards 8 and '14 which are ATtiny boards. In contrast to the ATtiny microcontrollers, the compatible ATmega

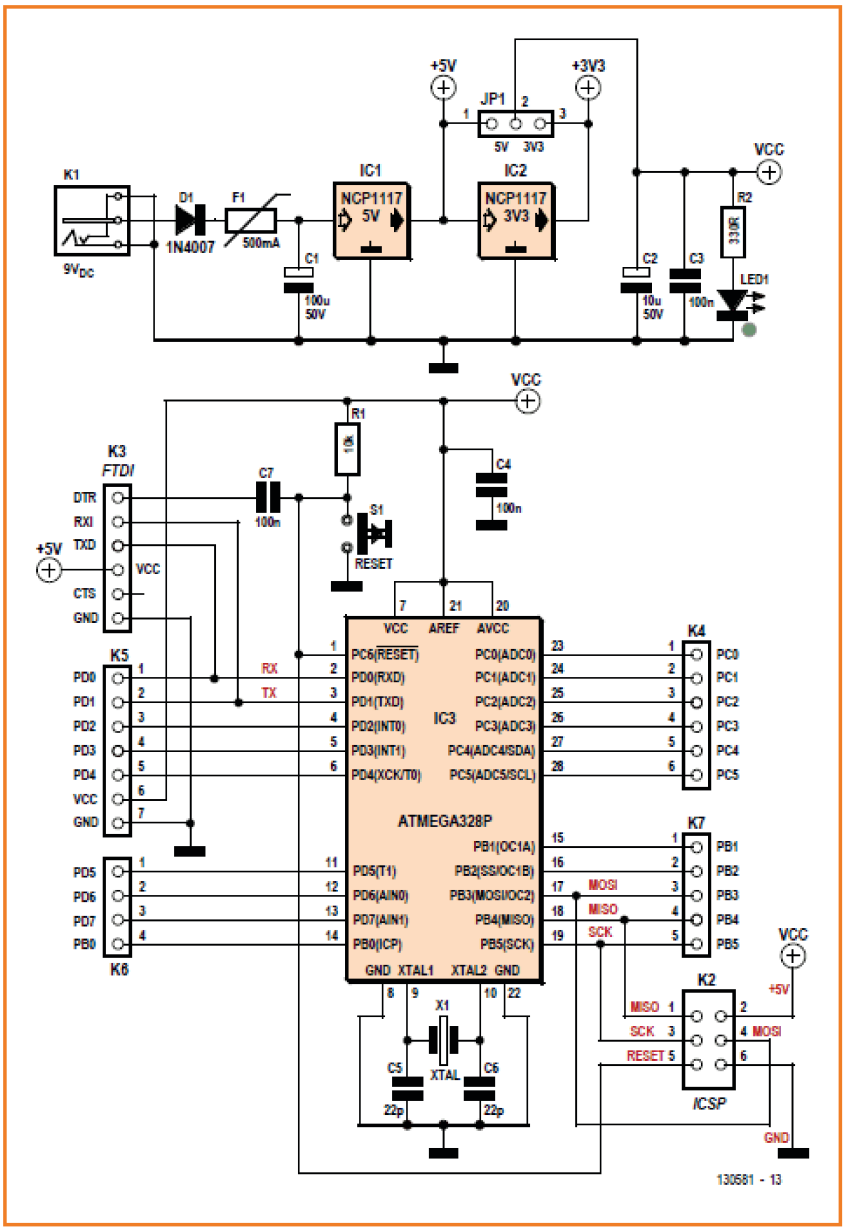


Figure 3.
T-Board 28 schematic. The largest board with the most powerful processor type ATMega328.

linker, assembler).

The pros: powerful and configurable IDE; open source; Windows/MacOS/Linux support, no code-size limitations; a strong community.

The cons: requires a plug-in to support Atmel MCUs; not a straightforward installation; does not support debugging natively.

Atmel Studio

This is Atmel's own offering [5], based on the Visual Studio platform. It of course supports all Atmel's MCUs including their ARM processors, so if you are focusing on Atmel only it's a good place to start.

The pros: support for all Atmel MCUs; advanced features including debugging and simulation; easy to install and get going; no code-size limitations.

The cons: Windows only; some may not like the Visual Studio interface; documentation could be stronger.

IAR Embedded Workbench [6]

An IDE targeted at the professional market, with a code-size limited version that can be used for free. In addition to supporting Atmel's AVR MCUs, it also supports microcontrollers from other major manufacturers which make it a good option if you want to work on multiple MCUs in a single tool.

The pros: support for all Atmel AVR MCUs, advanced features including debugging and simulation, a single professional tool for multiple manufacturers' MCUs.

The cons: free version restricted to 4 KB; expensive to buy; Windows only.

MCUs have a hardware UART and dedicated Tx/Rx pins. As a result, an FTDI header has been included for the 28-pin T-Board 28 only. This enables the board to communicate serially; connect a serial-to-USB convertor such as the Elektor FT232R USB/Serial Bridge [2] and you're able to communicate with a terminal program on your computer or program the microcontroller (assuming it has a bootloader). Serial communications can of course be implemented on the ATtiny in software, but with a lack of dedicated pins it was decided to exclude an FTDI header. The T-Board 28 additionally has a 2-way socket to accept an optional external quartz crystal. In line with Atmel's recommendations, 22-pF load capacitors are present. The decision was taken not to include these sockets for the smaller T-Boards, due to the limited number of I/O pins as well as the likelihood that the more common applications of these would not require the additional accuracy that an external crystal offers.

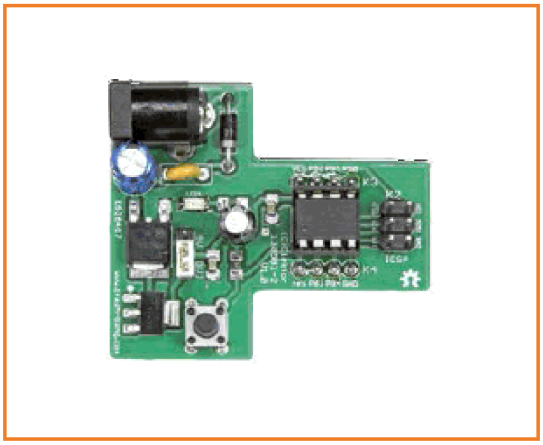


Figure 4.
T-Board 8 prototype. Small differences may exist with the final version supplied through the Elektor Store.

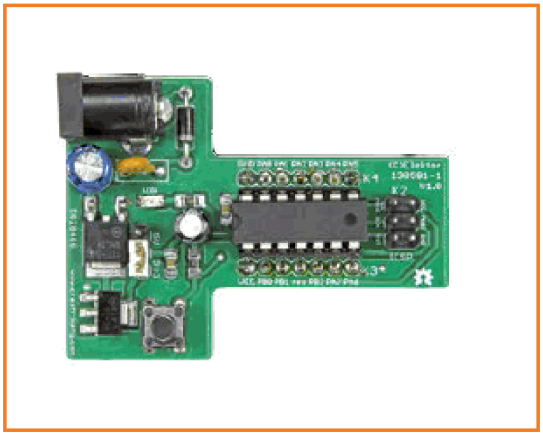


Figure 5.
T-Board 14 prototype. Small differences may exist with the final version supplied through the Elektor Store.

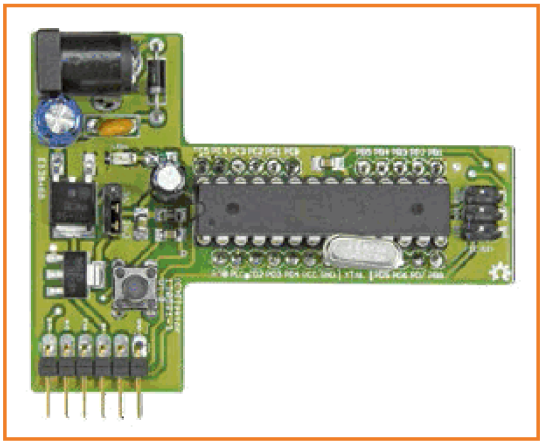


Figure 6.
T-Board 28 prototype. Small differences may exist with the final version supplied through the Elektor Store.

The 1-2-3 of T-Boards

As in the world of the Arduino, there are three steps to 'building' a T-Board project. However, now that we've moved onto working with a "raw" microcontroller, there is added complexity (and flexibility).

1. The first step, designing the physical project, is actually easier than with the Arduino. Snap the T-Board onto your breadboard, hook up your positive and negative power rails, and you can start placing components onto the breadboard without the need for additional jumper wires.

2. The second step, writing the code to control the MCU, can be as simple or complex as you like. You can either stick with the Arduino integrated development environment, or you can increase your flexibility by moving to a more fully-featured IDE such as Atmel Studio, Eclipse or IAR Workbench (refer sidebar: Choosing your IDE).

3. The third step is of course getting the compiled code onto the MCU. For an Arduino, this is taken care of by an on-board USB connector. However, there are limitations with this—for example using ATtiny MCU's. The T-Board provides the greatest degree of flexibility by including ICSP headers. This is the easiest way to flash your program onto the MCU (see **Inset**: Choosing your Programmer). You could use the FTDI header on the T-Board 28, but the ISP programmer is easier, simpler and faster.

A kickoff project

A blinking LED is the "Hello World" of the embedded world, and for our purposes a good place to start to illustrate the workings of the T-Board. For this project I'll be using Atmel Studio, as it is the simplest to get up and running (a quick download and install from the Atmel website). The same principles covered here would apply to your preferred IDE. We'll follow the three-step process discussed above.

Step 1: set up the breadboard

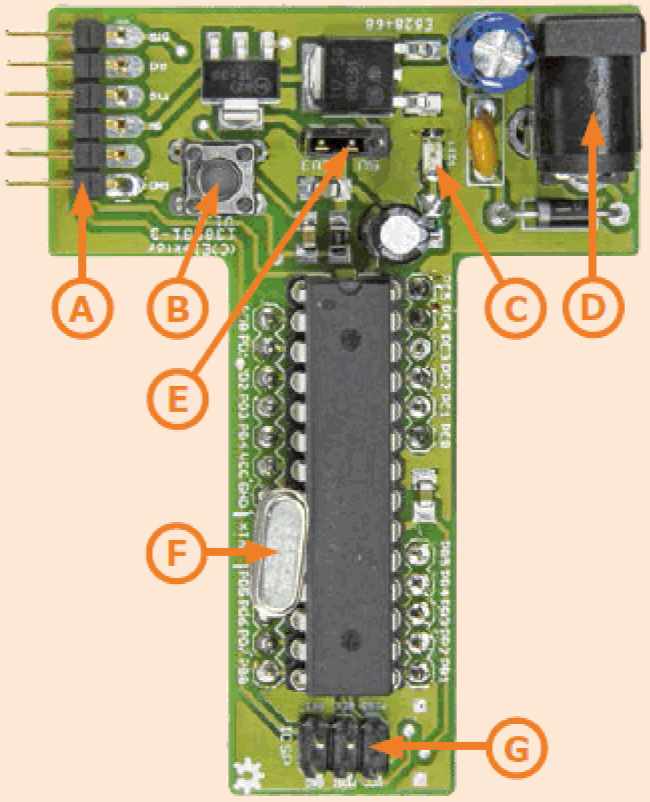
- Snap the T-Board onto the breadboard;
- Move the voltage selection jumper to the 5 V position;
- Connect a jumper wire between the GND pin and the breadboard's negative supply rail;
- Connect a resistor between an empty row on the breadboard and (depending on the T-Board being used):
  - PB0 on the T-Board 28
  - PA5 on the T-Board 14
  - PB4 on the T-Board 8
- Connect the anode of the LED to the resistor and the cathode to the negative supply rail.

Step 2: Write the Program

Create a new project in Atmel Studio, ensuring that you choose a 'GCC C Executable Project'. Select the correct device, depending on the T-Board being used:

- T-Board 28: ATmega328
- T-Board 14: ATtiny 84
- T-Board 8: ATtiny 85

Physical Layout and Function



example: T-Board 28

A. FTDI Connector:
Connect an FTDI breakout board for Serial communication over USB (T-Board 28 only)

B. Reset switch

C. Power LED

D. Power Connector:
A standard 2.1-mm center-positive jack (DC, max. 9 V)

E. Voltage Selection Jumper:
Allows the microcontroller to operate at either 5 V or 3.3 V

F. Crystal header pins:
Gives you the option of connecting an external crystal (T-Board 28 only)

G. ICSP Connector:
Connect an ISP programmer to program the microcontroller

Component List

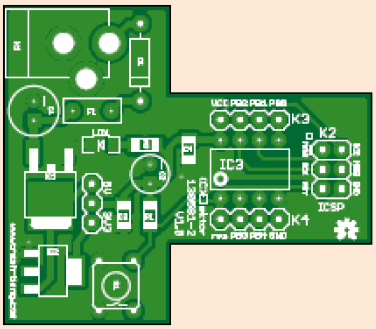
T-Board 8
Ref. no. 130581-92 (ATtiny45). Available ready-assembled from the Elektor Store

Resistors
R1 = 10kΩ 250mW 1%
R2 = 330Ω 250mW 5%

Capacitors
C1 = 100µF 50V radial
C2 = 10µF 50V radial
C3,C4 = 100nF 50V 10% X7R

Semiconductors
D1 = 1N4007
LED1 = SMD, green, 20mA
IC1 = NCP1117DT50G, 5V 1A regulator
IC2 = NCP1117ST33T3G, 3.3V 1A regulator
IC3 = ATTINY45-20PU, 8-bit MCU

Miscellaneous
K1 = DC barrel jack 2.1mm pin
K2 = 6-pin 2-row pinheader (2x3)
K3,K4 = 4-pin pinheader
S1 = switch, tactile, 24V 50mA, 6x6mm
F1 = 500mA PTC resettable fuse
JP1 = 3-pin pinheader
8-way DIL IC socket
jumper, 2-way, 0.1″ (2.54mm)
PCB # 130581-



Component List

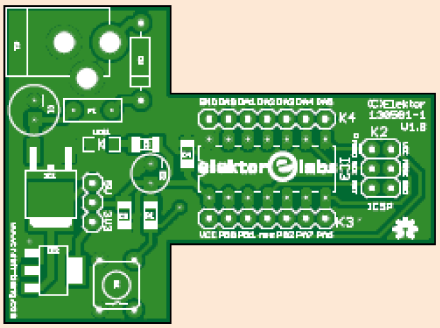
T-Board 14
Ref. no. 130581-91 (ATtiny84). Available ready-assembled from the Elektor Store

Resistors
R1 = 10kΩ 250mW 1%
R2 = 330Ω 250mW 5%

Capacitors
C1 = 100µF 50V radial
C2 = 10µF 50V radial
C3,C4 = 100nF 50V 10% X7R

Semiconductors
D1 = 1N4007
LED1 = SMD LED green 20mA
IC1 = NCP1117DT50G, 5V 1A regulator
IC2 = NCP1117ST33T3G, 3.3V 1A regulator
IC3 = ATTINY84-20PU 8-bit MCU

Miscellaneous
K1 = DC barrel jack 2.1mm pin
K2 = 6-pin 2-row pinheader (2x3)
K3,K4 = 7-pin pinheader, 1 x 7 pins
S1 = switch, tactile, 24V, 50mA, 6x6 mm
F1 = 500mA PTC resettable fuse
JP1 = 3-pin pinheader
14-way DIL IC socket
jumper, 2-way, 0.1″ (2.54mm)
PCB # 130581-1



Component List

T-Board 28
Ref. no. 130581-93 (ATMega328). Available ready-assembled from the Elektor Store

Resistors
R1 = 10kΩ 250mW 1%
R2 = 330Ω 250mW 5%

Capacitors
C1 = 100µF 50V radial
C2 = 10µF 50V radial
C3,C4,C7 = 100nF 50V 10% X7R
C5,C6 = 22pF, 50V, 1206

Semiconductors
D1 = 1N4007
LED1 = SMD, green, 20mA
IC1 = NCP1117DT50G 5V 1A regulator
IC2 = NCP1117ST33T3G 3.3V 1A regulator
IC3 = ATMEGA328P-PU 8-bit MCU

Miscellaneous
K1 = DC barrel jack, 2.1mm pin
K2 = 6-pin 2-row pinheader (2x3)
K3 = 6-pin pinheader, right angled
K4 = 6-pin pinheader
K5 = 7-pin pinheader
K6 = 4-pin pinheader
K7 = 5-pin pinheader
X1 = 2-way socket
S1 = switch, tactile, 24V, 50mA, 6x6mm
F1 = 500mA PTC resettable fuse
JP1 = 3-pin pinheader
28-way 300mil width DIL IC socket
Jumper, 2-way, 0.1″ (2.54mm)
PCB # 130581-3

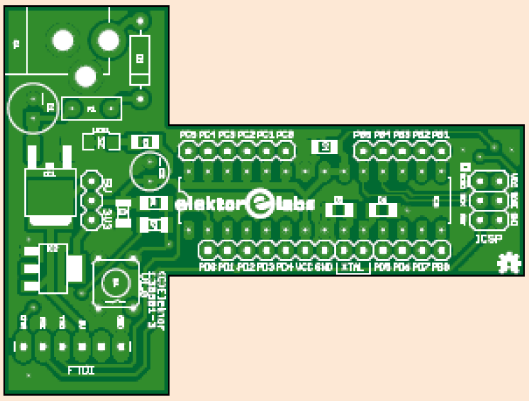


Figure 7. Component overlays of T-Board 8, T-Board 14, and T-Board 28.
Note that all T-Boards are available ready-assembled from the Elektor Store www.elektor.com.

Then enter the code from **Listing 1** into Atmel Studio. This code is far from optimized, but in retaining simplicity it achieves what we need for our purposes here. If you come from an Arduino background, you probably won't recognize some of the code—if so, the *Microcontroller BootCamp* series of articles starting in the April 2014 edition of Elektor magazine is compulsory reading! If you are not using the T-Board 28 then change the above code to refer to the LED pin, as summarized in **Table 1**.

Once the code is entered, it needs to be compiled. Ensure that the Configuration Manager is set is set to "release", then press F7 to build.

Step 3: Flash the program to the T-Board

Connect the ISP programmer you'll be using to the T-Board and PC

Select the ISP Programmer that you will be using, by select the **Project** menu, then **...properties**

Table 1. Changes to T-Board 28 program code to suit T-Board 8, T-Board 14.

Existing	T-Board 14	T-Board 8
DDRB	DDRA	DDRB
DDB0	DDA5	DDB4
PORTB	PORTA	PORTB
PORTB0	PORTA5	PORTB4

On the **Tool** tab, select the **debugger/programmer** you're using.

Upload the program to the T-Board: Click on **Debug** menu, then **Start without Debugging**

The LED should start blinking.

If you like you can now disconnect the T-Board from the ISP programmer, and connect it to a 9-V battery to operate in stand-alone mode.

Hopefully you'll agree that this was a great deal simpler than laying a full microcontroller breadboard out.

Listing 1. T-Boards LED Blinker.

```
/*
 * T_Board_Blink.c
 *
 * Created: 24/05/2014 11:54:26
 *  Author: Andrew Retallack, Crash-Bang Prototyping
 *
 */

#define F_CPU 16000000UL //We are running at 16MHz. Used to time the delay

#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
  //Configure the LED port
  DDRB |= (1<<DDB0);  //Set Pin PB0 as an output pin

    while(1)
    {
         PORTB |= (1<<PORTB0);  //Turn the LED on, by making PB0 high
     _delay_ms(1000);     //Delay 1 second
     PORTB &= ~(1<<PORTB0);  //Turn the LED off, by making PB0 low
     _delay_ms(1000);     //Delay 1 second
    }
}
```

The Author

With a background in information systems and software development, Andrew Retallack discovered the joy of using electronics to interact with the physical world a few years ago. He is enthusiastic about sharing his learning with others who do not have formal electrical engineering training, and focusses on AVR and MSP430-based sensing, RF and automation projects.

Going Further

The T-Board family was designed to provide greater flexibility than the Arduino. Of particular interest to the author was the ability to easily and relatively cost-effectively optimize the power consumption of projects. A future article will discuss ways to use the T-Board to measure the current consumption of a project, and then how to reduce this by altering the input voltage, using low power modes and interrupts, and changing the processor clock speed.

(130581)

Choosing your programmer

An ISP is an In-System Programmer, a hardware tool that enables you to flash (or download) your compiled code onto a microcontroller. There are a wide range of ISPs available, but here are three options that come in at the hobbyist end of the market. If you know of other options, let us know.

Arduino as an ISP

You may already know that an Arduino can function as an ISP. Simply connect four of the Arduino pins to the ICSP header on the T-Board, as well as power and ground, and you're good to go. There's a detailed tutorial available online to step you through the process [7]. There is of course no cost involved if you already own an Arduino; if you don't own an Arduino it's recommended that you look at buying a dedicated programmer. Admittedly the disadvantage to using an Arduino is having to wire the T-Board up every time you want to program it—you may want to build a cable with the ICSP header to speed this up.

USBTinyISP

This is an open source collaborative effort, and actually uses an ATtiny MCU as the programmer. It is a great low-cost option, and if you are interested you can build it yourself from parts (also available as a kit). The only disadvantage is that it is not natively supported by Atmel Studio, as it uses the AVRDude software to control the flashing of the controller. This does mean a few extra steps [8] to get it working as well as some reduced functionality (e.g. Setting of Fuses), but it could be worth the savings in cost.

AVRISP mkII

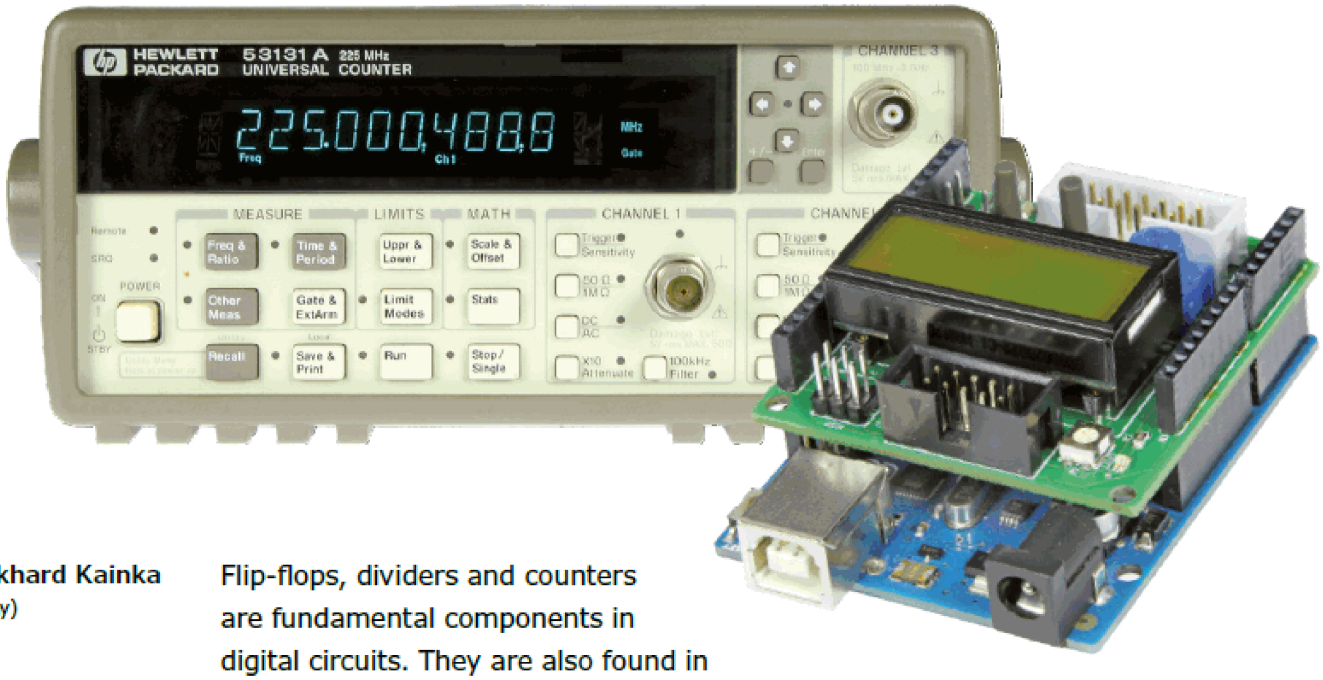
Atmel's own ISP, the AVRISP mkII [9], comes in as the most expensive of the three listed here, but at $34 (at time of print) it's still within reach. The benefit of this tool is that it is natively supported by Atmel Studio—if that's your IDE of choice. However, it requires some additional configuration to get it up and running with other IDEs, and does not seem to be supported by IAR Embedded Workbench.

Web Links

[1] www.crash-bang.com/resource/breadboard-arduino/
[2] www.elektor.com/ft232r-usb-serial-bridge-bob-110553-91
[3] http://arduino.cc/en/Main/Software
[4] www.eclipse.org/
[5] www.atmel.com/Microsite/atmel_studio6/
[6] www.iar.com/Products/IAR-Embedded-Workbench/AVR/
[7] www.crash-bang.com/resource/bootload-atmega328/
[8] www.crash-bang.com/using-usbtiny-with-atmelstudio/
[9] www.atmel.com/tools/avrispmkii.aspx
[10] www.elektor-magazine.com/130581

Microcontroller BootCamp (5)

Using timers



by **Burkhard Kainka**
(Germany)

Flip-flops, dividers and counters are fundamental components in digital circuits. They are also found in microcontrollers, where they have a wide range of uses: here we look at a few.

One of the longest chapters in the ATmega328 data sheet is the one that describes its three timers. The timers can be used in so many different ways that we only have space here to look at a small fraction of the possibilities. The main application areas are in measuring time intervals and frequencies, and in generating various signals including PWM output.

Measure those microseconds

The need to measure time intervals often crops up in electronics. For example, we might want to know how long it takes to output some data to an LCD module: is it milliseconds or microseconds? Without some inside knowledge, the only way is to measure it. The ATmega328 has a suitable module already on board in the form of Timer1, which has a resolution of 16 bits. Now, all the timer module does is simply act as a counter of regular events. In this case, with a 16-bit counter, the maximum count is 65535, after which the counter returns to zero: this is called 'overflow'. If things are arranged so that the counter increments once a microsecond we can measure time intervals of up to 65535 µs. Between the crystal oscillator and the clock input of the counter there is a programmable prescaler (similar to the arrangement for the A/D converter) that divides down the clock frequency. If a 16 MHz crystal is used then the required division ratio to obtain a 1 MHz clock for the timer is 16. So our first attempt reads

```
Config Timer1 = Timer , Prescale = 16
```

but unfortunately this does not work. The error message we get is 'Prescale value must be 1, 8, 64, 256 or 1024', and the data sheet of the ATmega328 confirms the problem. So, we shall set the prescaler ratio to 8 and obtain an input clock to the timer of 2 MHz and a maximum interval measurement of 32767.5 µs. At any time the 16-bit register that holds Timer1's value can be read or a new value can be written to it. This makes our example very straightforward. Before outputting the data to the LCD module we set the counter to zero ('Timer1 = 0'). After outputting the data we read the counter value ('D = Timer1') and we have the result of the measurement in units of 0.5 µs. To convert to microseconds we simply divide the result by 2. **Listing 1** shows the complete program.

As in all the programs we shall look at here, the result is displayed on the Arduino shield LCD module we described in the previous installment in this series, and simultaneously the result is output to the terminal using a 'Print' command. The program will work even if the display is not connected: Bascom does not check whether the display has actually received the data it sends to it. The first result we see is 'Timer 1 = 0 us': this happens because at this point, before the first measurement, the variable D has not been set. The second result looks rather more interesting, giving the time taken to output the first result to the LCD: 'Timer 1 = 17105 us'. Now in this case four additional digits have had to be sent to the display, and so we would expect the process to take a little longer. And indeed it does, the third result being 21933 µs. The results now stabilize, with variations between successive readings of at most one microsecond.

So now we know exactly how long a complex process like writing a string to the LCD takes. The precision and repeatability we have seen would be unimaginable on a Windows or Linux machine, since these complex operating systems do not handle real-time functions well: there is always some process running in the background that makes it impossible to predict exactly how long some action will take. Under Bascom things are different: the processor executes exactly the code you tell it to execute and nothing more. Moreover, being so close to the hardware, some commands such as setting an output bit execute in less than a microsecond. Outputting to the LCD takes around a millisecond per character because Bascom allows plenty of time for the controller in the module to accept the data. If you take a look at the E signal on the LCD (port pin PD3, Arduino pin 3) using an oscilloscope you will see the 1 ms delays.

Measuring the period of a signal

A small modification to our program (see **Listing 2**) allows it to measure the length or period of a pulse. We use input PC0 (Arduino A0: see **Figure 2**), and we will measure the time between two rising edges of the input signal. We need two sampling loops to detect an edge reliably: first we wait until the input reads as a zero, and then we wait until it reads as a one. This will detect

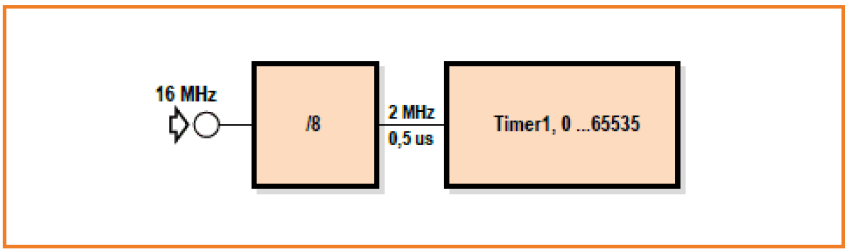


Figure 1.
Time measurement using Timer1.

Listing 1. Measuring the time taken to output to the LCD.

```
'-------------------------------------
'UNO_Timer1.BAS  Timer1 0.5 us
'-------------------------------------
...

Dim D As Word
Config Timer1 = Timer , Prescale = 8
'Clock 2 MHz

Do
  Print "Timer1 = ";
  Print D;
  Print " us"
  Timer1 = 0
  Locate 1 , 1
  Lcd "Timer1 ="
  Locate 2 , 1
  Lcd D
  Lcd " us"
  D = Timer1
  D = D / 2
  Waitms 1000
Loop
```

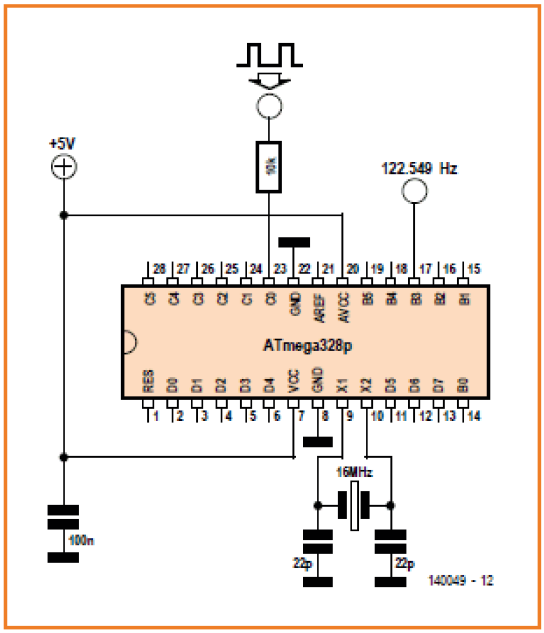


Figure 2.
Measuring the period of a signal.

the first rising edge, and we set the timer register to zero. We now wait for the second edge, and read the result from the timer.

If a finger is touched on the input pin, the microcontroller will receive a signal picked up from the mains supply, which will be at 60 Hz in the USA and 50 Hz in most other countries. We would expect a result of around 16667 µs or 20000 µs respectively. In a real experiment (carried out in Europe) a reading of 20030 µs was obtained, a little on the slow side. Since the mains supply in most of mainland Europe is phase-synchronous, we can only conclude that perhaps too many power stations had been turned off or the sun was not shining brightly enough or the wind not blowing strongly enough. Perhaps it might be worth turning a couple of lights off and trying again...

In order that we have a reliable signal to measure, the program includes its own clock source, taking advantage of the PWM outputs. However, there is a potential problem. PWM1A and PWM1B cannot be used because we have already committed Timer1 for making the measurement itself. That leaves us with Timer0 and Timer2, each of which can also drive two PWM outputs. However, these extra outputs mostly appear on Port D and would therefore interfere with the LCD interface. An exception is PWM2A which is on port pin PB3 (Arduino pin 11), and so this is the one we use to output our test signal. Timer2 has only an 8-bit counter but if we set its prescaler ratio to 256, we can obtain an output frequency of 16 MHz / 256 / 255 / 2 = 122.549 Hz and hence an output period of 8.16 ms. Connecting PB3 to PC0 lets us measure this signal, and the reading we get is 8160 µs. Result!

Square wave generator, 125 Hz to 4 MHz

A variable-frequency signal generator is often a

Listing 2. Measuring the period of a signal in microseconds.

```
'----------------------------------------
'UNO_Timer2.BAS  Timer1,  0.5 us
'----------------------------------------
...

Dim D As Word

Config Timer1 = Timer , Prescale = 8  'Clock 2 MHz
Config Timer2 = Pwm , Prescale = 256 ,
    Compare A Pwm = Clear Up
Ddrb = 255
Pwm2a = 128

Do
  Do
  Loop Until Pinc.0 = 0
  Do
  Loop Until Pinc.0 = 1
  Timer1 = 0
  Do
  Loop Until Pinc.0 = 0
  Do
  Loop Until Pinc.0 = 1
  D = Timer1
  D = D / 2
  Locate 1 , 1
  Lcd "Timer1 ="
  Locate 2 , 1
  Lcd D
  Lcd " us    "
  Print "Timer1 = ";
  Print D;
  Print " us"
  Waitms 1000
Loop
```

Listing 3. Timer1 used as a square wave generator.

```
'----------------------------------------
'UNO_Timer3.BAS  B1 Fout 250 Hz...4 MHz
'----------------------------------------
...

Dim D As Long
Dim F As Long
Dim N As Byte

Config Timer1 = Pwm , Prescale = 1 , Pwm = 10 ,
    Compare A Pwm = Clear Up

Tccr1a = &B10000010    'Phase-correct PWM, Top=ICR1
Tccr1b = &B00010001    'Prescaler=1

D = 8000
Do
  N = Ischarwaiting()
  If N = 1 Then
    Input F
    D = 8000000 / F
    If D > 64000 Then D = 64000
    If D < 2 Then D = 2
  End If
  If S1 = 0 Then
    D = D + 1
    If D > 100 Then D = D + 1
    If D > 1000 Then D = D + 100
    If D > 10000 Then D = D + 1000
    If D > 64000 Then D = 64000
  End If
  If S2 = 0 Then
    If D > 2 Then D = D - 1
    If D > 100 Then D = D - 10
    If D > 1000 Then D = D - 100
    If D > 10000 Then D = D - 1000
    If D > 64000 Then D = 64000
  End If
  Locate 1 , 1
  F = 16000000 / D
  F = F / 2
  Lcd F
  Lcd " Hz      "
  Icr1 = D
  Ocr1a = D / 2
  Waitms 50
Loop
```

useful tool to have. At 1 MHz it might be used to test a frequency counter, or at 440 Hz it might be used to tune a violin. The program shown in **Listing 3** covers the whole range from 125 Hz to 4 MHz. Timer1 is used as a frequency divider and the signal is output on the PWM1A pin (B1, Arduino pin 9: see **Figure 3**). This is an example of an application where we are going a little beyond the standard uses of Bascom, and in particular its built-in initialization functions are not suitable for our purposes: we have to program a few registers directly. I borrowed some ideas from Roger Leifert's DCF simulator [1], and he in turn borrowed from the code in Martin Ossman's SDR course [2], which is written in C. We use a variant of PWM output mode where we adjust the frequency while trying to maintain a duty cycle of approximately 50%. Register LCR1 is loaded with the desired division ratio. If, for example, this value is 100, then the output frequency will be 16 MHz / 100 / 2 = 80 kHz. To ensure that the output is a symmetric square wave we need to load register OCR1A with the value 50.

The program can be controlled (simultaneously) in two different ways: over the serial port using a terminal emulator program or using buttons. When pressed, the buttons increase or decrease the division ratio, and the program calculates the resulting frequency and displays it. A brief press of a button changes the ratio in small steps, while a longer press causes the ratio to change

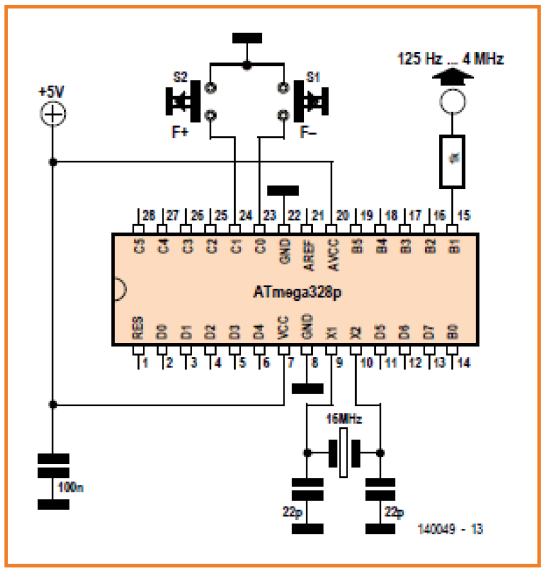


Figure 3.
Adjustable square wave generator.

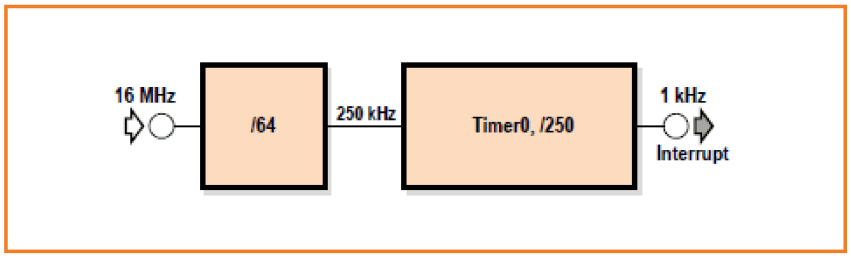


Figure 4.
Producing 1 kHz from 16 MHz.

(almost) continuously. Since it would be rather tedious to step through all the possible ratios in this way (there are more than 65000 of them), the amount of increment changes depending on the frequency range. This means that many frequencies will only be approximated rather than generated exactly. The output frequency is displayed in Hertz and the numbers involved are such that the calculations must be done not using word variables but with 'long' (32-bit) quantities: this applies both to the frequency F and to the division ratio D.

Listing 4. Counting seconds exactly using an interrupt.

```
'-----------------------------------
'UNO_Timer4.BAS
'Timer0-Interrupt, Seconds
'-----------------------------------
...

Dim Ticks As Word
Dim Seconds As Word
Dim Seconds_old As Word

Config Timer0 = Timer , Prescale = 64
On Ovf0 Tim0_isr
Enable Timer0
Enable Interrupts

Do
  If Seconds <> Seconds_old Then
    Print Seconds
    Seconds_old = Seconds
    Locate 1 , 1
    Lcd Seconds
  End If
Loop

Tim0_isr:
  '1000 µs
  Timer0 = 6
  Ticks = Ticks + 1
  If Ticks = 1000 Then
    Ticks = 0
    Seconds = Seconds + 1
  End If
Return

End
```

It is perhaps not completely obvious how the program manages to respond both to button presses and to input on the serial port. If we write simply 'Input F' then the program will wait at this point until a value is entered, and will not respond to the buttons. We therefore have to check first whether there is a character available on the serial interface: the function IsCharWaiting() returns a value of 1 if there is at least one character available and zero otherwise. In our case, if one character arrives on the serial interface then we know that there are more to come, and we can safely read in a new value for F. From F we can calculate the required division ratio, in many cases obtaining an exact result. At 440 Hz there is only negligible error, although at 549 kHz, for example, only a rough approximation is available: the program chooses a division ratio of 14, which results in an output frequency of 571428 Hz. The smallest division ratios give rise to the highest frequencies, namely 4 MHz, 2.666666 MHz, 2 MHz, 1.6 MHz, 1.333333 MHz and 1 MHz. If these are useful to you, then you could build the design as a self-contained unit.
The steep edges on the signal on output PB1 mean that it contains harmonics well into the VHF range. This means it is all too easy for the signal generator to become a source of radio interference. For example, if you connect the oscilloscope probe to the output but forget to connect the ground clip close by then a large and rather effective VHF loop antenna can be created via the ground connection to the USB port and then through the PC's power supply and the mains. Unsurprisingly this can disrupt reception on nearby FM radios and hence relations with your neighbors! To keep on the right side of the EMC regulations (and your neighbors) it is necessary to use either a screened cable, or a resistor close to the signal generator's output to reduce the amplitude of the higher harmonics. A resistor of 1 kΩ in conjunction with a cable capacitance of 30 pF forms a low-pass filter with a cutoff frequency of 5 MHz. The edges of the signal are

smoothed considerably, and VHF interference is reduced by around 20 dB.

Timer interrupts

In the previous installment of this series we looked at an example program that generates a one-second clock. The timing used a simple 'Waitms 1000' command. Now this approach does not give an exact result, as the other parts of the program such as the infinite loop and the code that generates the output all take time. The problem can be avoided using a timer interrupt: that is, allowing a certain part of the code to execute exclusively under control of a timer. It works as follows. A hardware timer counts away, completely independent of other activity in the microcontroller. When it reaches its maximum count and resets to zero ('overflow') the program executing in the foreground is interrupted and a special piece of code called an 'interrupt service routine', or ISR, is called. Within this piece of code we can carry out any actions that need to happen at exactly-specified time intervals. It makes no difference how long the ISR itself takes to execute as the hardware timer is continuing to count: the only thing that matters is that the routine completes before the next time the timer overflows and triggers the interrupt again.

For the job of generating a precise one-second clock Timer1 is overkill: the eight-bit resolution of Timer0 is plenty for this application. We will arrange for the timer to overflow and hence generate an interrupt every 1000 µs. The interrupt service routine, which, as it is triggered from Timer0, we have called 'Tim0_isr:', will thus be called every millisecond. The colon means that 'Tim0_isr:' will be interpreted as a label, marking a point in the code which can be jumped to. The command 'On Ovf0 Tim0_isr' tells the microcontroller to jump to this label whenever there is an overflow ('Ovf') on Timer0. The interrupt service routine must finish with a 'Return' command: after that point the interrupt routine can be called again.

The example program shown in **Listing 4** initializes Timer0 with a prescale ratio of 64, which means it increments at 250 kHz (see **Figure 4**). If we did not take any further steps the timer would overflow on every 256th clock, as the counter is eight bits wide. To arrange for an overflow every 250 clocks, the first thing the interrupt service routine does is load the counter with the value 6. The result is that the routine is called exactly every millisecond. The word variable 'Ticks' is incremented by one on each timer overflow. When it reaches 1000 the variable 'Seconds' is incremented. The variables 'Seconds' and 'Ticks' can be read from the main program code. In this example the program outputs the number of seconds since it was started, both to the LCD and to the terminal.

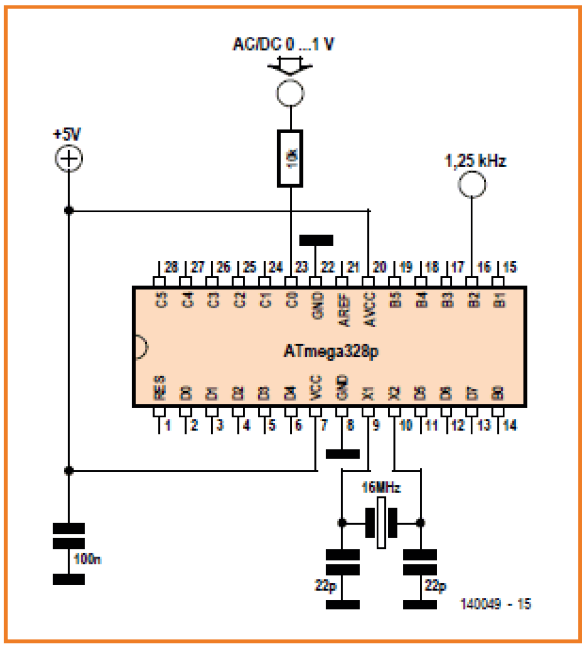


Figure 5.
AC and DC voltage measurement.

To ensure that the interrupt service routine is actually called, the corresponding interrupt (the Timer0 overflow interrupt) must be enabled. This is done with the line 'Enable Timer0'. Interrupts must also be globally enabled, using the command 'Enable Interrupts'. The complementary command 'Disable Interrupts' prevents all interrupts from occurring.

Averaging analog readings

Analog readings often have mains hum, at 50 Hz or 60 Hz depending on the local frequency, superimposed on them. One way to try to remove this is to take the average of a number of consecutive samples: see **Listing 5**. If readings are averaged over an integer number of mains cycles (that is, over a multiple of 20 ms or 16.667 ms respectively), the effect is to create a null at that frequency. In other words, the hum will be significantly attenuated.

Again in this example we use a timer interrupt to ensure accurate timings. We will take the average of 500 consecutive readings from ADC0 (see **Figure 5**), which a total of 400 ms. Now this is

Listing 5. Averaging within a timer interrupt.

```
'-----------------------------------------------
' UNO_Timer5.BAS  Timer1-Interrupt, ADC0 average
'-----------------------------------------------
...

Dim Ticks As Word
Dim Ad As Word
Dim Ad0 As Long
Dim Ad0_mean As Long

Config Adc = Single , Prescaler = 32 , Reference = Internal
Config Portb.2 = Output

Config Timer2 = Timer , Prescale = 64
On Ovf2 Tim2_isr
Enable Timer2
Enable Interrupts

Do
  Disable Interrupts
  Ad0_mean = Ad0_mean * 2443       'AC
  'Ad0_mean = Ad0_mean * 1100      'DC
  Ad0_mean = Ad0_mean / 1023
  Ad0_mean = Ad0_mean / 500
  Print Ad0_mean
  Locate 1 , 1
  Lcd Ad0_mean
  Lcd " mV     "
  Enable Interrupts
  Waitms 1000
Loop

Tim2_isr:
  '800 µs
  Timer2 = 56
  Portb.2 = 1
  Ticks = Ticks + 1
  Ad = Getadc(0)
  Ad0 = Ad0 + Ad
  If Ticks >= 500 Then
    Ticks = 0
    Ad0_mean = Ad0
    Ad0 = 0
  End If
  Portb.2 = 0
Return

End
```

exactly 20 periods at 50 Hz and 24 periods at 60 Hz and so in either case we are averaging over an integer number of mains cycles. The period between conversions is 800 µs, and to generate this timing we use Timer2, again with a prescale ratio of 64. Each time Timer2 overflows it is reloaded with the value 56, and so the next overflow occurs exactly 200 clocks later.

Eight hundred microseconds is long enough to carry out one conversion (or even several) and accumulate the results. The calls to the interrupt routine and hence the individual readings are counted in the variable 'Ticks', and after 500 readings have been accumulated the sum in variable 'AD0' is copied to the variable 'AD0_mean'. The foreground code can read this variable and send the result to the terminal.

It is sound practice to use an oscilloscope to check that the interrupt routine is running as expected. Is it really being executed every 800 µs, that is, at 1.25 kHz? How long does it take to service the interrupt? A simple trick helps to see what is going on: at the start of the interrupt service routine set a port pin high, and at the end take it low again. In this example we use port PB2, which is connected to LED 2. The oscilloscope does indeed show a pulse every 800 µs lasting about 60 µs, and so there is nothing to worry about. In other cases, however, it can happen that so much is packed into the interrupt service routine that the main program never gets executed, and it is not always obvious what is happening. Furthermore, when interrupts are used, the timing of the main program is no longer so easily predictable; a good rule of thumb is to ensure that no more the 50 % of the microcontroller's time is spent in interrupt service routines.

The averaging process is so good at removing the mains hum component of the signal that it can be used to measure AC voltages using half-wave rectification. The A/D converter already performs half-wave rectification, in that it only measures positive voltages and all negative voltages read as zero. If an AC voltage is applied to the A/D converter input via a 10 kΩ series protection resistor then the converter will only see the positive half-cycles. The program will then average these readings, with the result that, in the case of a square wave input, the average of these half cycles is half the DC voltage with the same effective (RMS) value as the input. In the case of a sinusoidal input there is a further factor of pi/2 = 1.571: in other words, the calculated

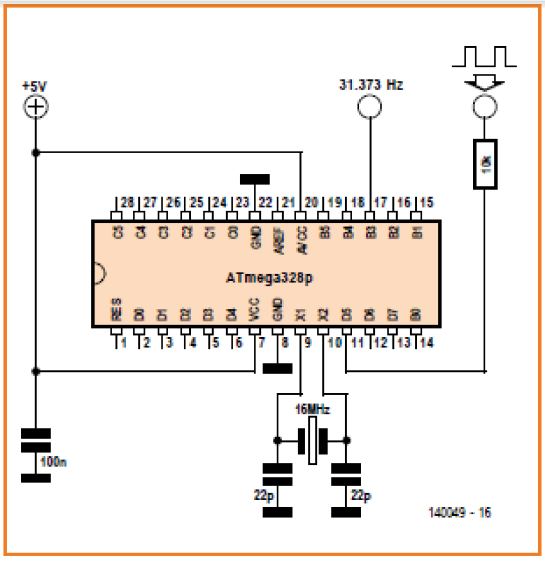


Figure 6. Frequency meter with test output.

average is 90.03 % of the true RMS value. These factors can be taken into account in calculating the displayed reading in millivolts. For pure DC measurements the fact that the reference voltage is 1100 mV means the correction factor is 1100. For AC voltage measurements the factor should be 2443, and readings will be correct up to a peak input voltage of 1.1 V. The procedure also works at other frequencies, and in fact voltages at any frequency from 50 Hz to 50 kHz can be measured, meaning that the set-up can be used as a wide-bandwidth millivoltmeter in audio and other applications.

Frequency measurement

In the examples we have looked at so far the timer has received its clock pulses from within the processor, either directly from the processor's clock or via a divider. However, it is possible to clock the timer using an external signal, in which case the timer behaves as a pulse counter. Timer1 in an ATmega328 clocked at 16 MHz can reliably count external pulses at a frequency of up to 4 MHz. Unfortunately, the input lies on port pin PD5 (see **Figure 6**) and so we cannot use the LCD module as well. We could alternatively use an external LCD module controlled over a serial interface, but for now we will simply send our results to a terminal emulator running on a PC. To use Timer1 as part of a frequency counter (see **Listing 6**) we also need to measure the

Listing 6. Frequency measurement up to 4 MHz.

```
'--------------------------------------
'UNO_Timer6.BAS  Frequency 0...4 MHz
'--------------------------------------
...

Dim Lowword As Word
Dim Highword As Word
Dim Ticks As Word
Dim Freq As Long

Config Timer0 = Timer , Prescale = 64
On Ovf0 Tim0_isr
Enable Timer0

Config Timer1 = Counter , Edge = Falling , Prescale = 1
On Ovf1 Tim1_isr
Enable Timer1

Config Timer2 = Pwm , Prescale = 1 , Compare A Pwm = Clear
Up
Pwm2a = 128        'B3: 31373 Hz
Enable Interrupts

Do
  Print Freq;
  Print " Hz        ";
  Print Chr(13);
  Waitms 1000
Loop

Tim0_isr:
  '1000 µs
  Timer0 = 6
  Ticks = Ticks + 1
  If Ticks = 1 Then
    Timer1 = 0
    Highword = 0
  End If
  If Ticks = 1001 Then
    Lowword = Timer1
    Freq = Highword * 65536
    Freq = Freq + Lowword
    Ticks = 0
  End If
Return

Tim1_isr:
  Highword = Highword + 1
Return
End
```

Listing 7. Text output on the LCD.

```
'-----------------------------------
'UNO_Display.BAS  COM input B0
'-----------------------------------
...

Dim Text1 As String * 16
Dim Text2 As String * 16

Open "comb.0:9600,8,n,1" For Input As
#2
'Software COM input at B0

Do
  'Input Text1
  Input #2 , Text1
  Locate 1 , 1
  Lcd Text2
  Text2 = Text1 + "        "
  Locate 2 , 1
  Lcd Text2
Loop

End
```

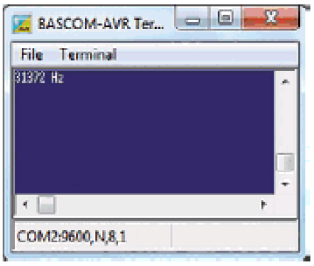


Figure 7.
The frequency displayed using the terminal emulator.

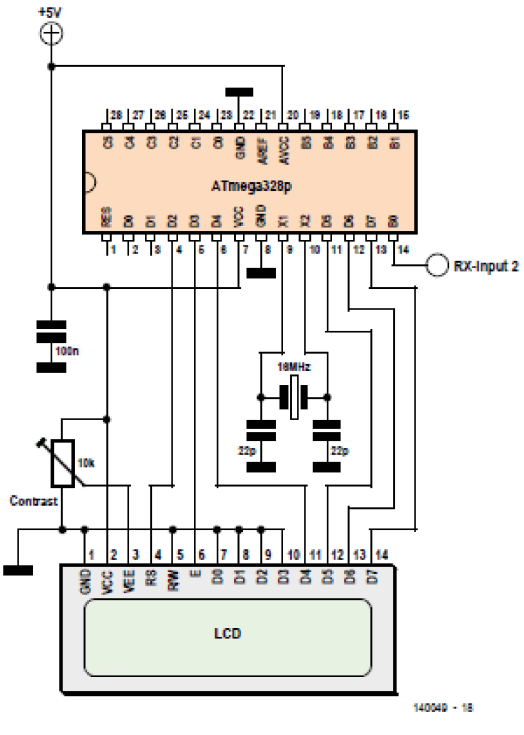


Figure 8.
The LCD terminal.

gate time accurately. To do this we use a second timer and two interrupts. So that we can measure frequencies above 65535 Hz, the interrupt service routine Tim1_isr is called whenever it overflows to increment the variable 'Highword'. Timer0 is responsible for providing a gate time of exactly one second. When the variable 'Ticks' is equal to one Timer1 is reset and counting starts. Exactly 1000 ms later the current counter value in Timer1 is read into the variable 'Lowword' and then the frequency is calculated from this and the value in 'Highword'. The foreground code can now output the result as a frequency in Hertz.

If we configure Timer1 as an ordinary timer with a 16 MHz clock (using the command 'Config Timer1 = Timer , Prescale = 1') then we should see a reported frequency of exactly 16000000 Hz. With the timer configured as a counter, however, the maximum increment rate is limited to a quarter of the processor clock, because the state of the input pin is only sampled at a limited rate. If we try an input frequency of 6 MHz we find that approximately every other pulse is lost, giving a reading of around 3 MHz. At frequencies up to a little over 4 MHz, however, the counter is very accurate.

Observe one special aspect of the 'Print' commands. Normally Bascom terminates each print command by emitting Chr(13) and Chr(10), which makes it easy to send the output of one Bascom program to the input of another. However, in the receive direction only the character Chr(13) is expected, which means that we can suppress the line feed character (the Chr(10)) and send just the carriage return (the Chr(13)). To display the results we can use Bascom's own terminal emulator: the effect is that each new reading

overwrites the old on the same line rather than beginning a new line for each: see **Figure 7**.

External display

A simple solution to the problem of not being able to use the display directly is to use another Arduino. One, with an Elektor shield fitted, functions as the display module, while the other acts as the frequency counter, sending its results over a serial interface to the first. **Listing 7** shows the code for a simple display with scrolling output. The last two lines are always shown.

The program offers two possibilities for receiving serial data. The command 'Input Text1' (commented out) uses the normal serial RX input on D0. This input is connected to the USB interface on the Uno board via a 1 kΩ series resistor. This signal can be overridden using a low-impedance drive, as for example happens when the RX pin on the display unit is connected directly to the TX pin of the transmitting Uno (the one carrying out the frequency measurement). The disadvantage is that communication during the next program upload using the bootloader will be disturbed. It is easy to forget this, which can lead to a lot of head-scratching when you next make a change to the program!

The second possibility takes advantage of Bascom's ability to implement a serial port in software using any desired port pin. Here we have chosen PB0 (Arduino pin 9: see **Figure 8**) and use the command 'Input #2, Text1'. The text is received just as reliably, and there is no interference with program upload.

Web Links

[1] Roger Leifert, 'DCF Tester', Elektor June 2014, www.elektor-magazine.com/130571

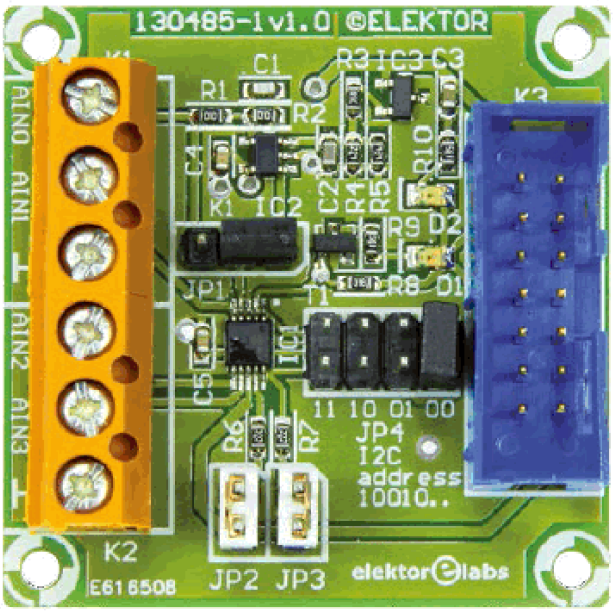
[2] Martin Ossmann, 'AVR Software Defined Radio', Elektor April 2014, www.elektor-magazine.com/100181

16-bit Data Logger

An ADC module for Arduino, Elektor Linux Boards and more

By **Jens Nickel**
(Elektor Germany)

To make accurate voltages measurements you need an A/D converter (ADC) chip which has good resolution. The folks at Elektor Labs have developed a board containing a four-channel 16-bit ADC. It's no coincidence the board uses a Gnublin/ EEC connector which makes it compatible with the Elektor's Linux board, Xmega Webserver and the new Extension shield for Arduino. A universal C library is included to make integration a walk in the park.



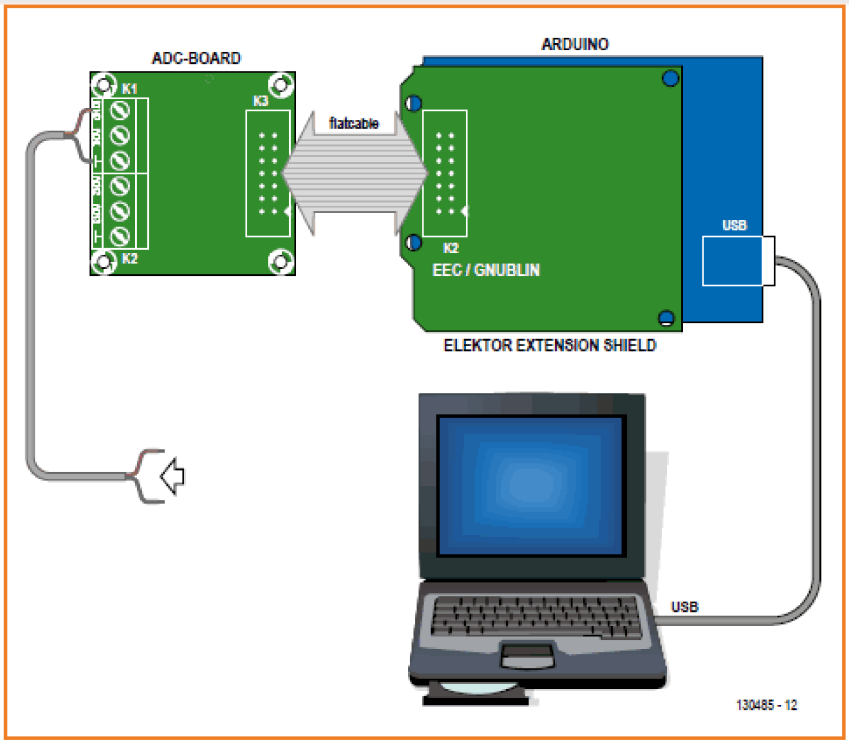
This project is a good example of where our web-based project platform Elektor.Labs [1] has provided the spark for new ideas. Kurt Diedrich, an author well known to us, has recently been busy experimenting with an Arduino Uno to expand his knowledge of microcontroller firmware. It didn't take too much encouragement to persuade Kurt to share his experiences with a wider audience via Elektor.Labs. One of the designs he came up with is a data logger where an Arduino Uno measures a signal level and sends it to a PC [2]. Kurt's previous experience with PC programming and graphical user interface design could now be put to good use: using the programming language called Processing he designed a Tool that not only represents the sampled signal but can also performs spectral analysis. The complete setup is ideally suited for processing the reception of low-frequency electromagnetic signals to be discussed in a follow up article describing an ELF Receiver (ELF = extremely low frequency).

Hunting for ELF signals

At Elektor.Labs you can read more details of the original project [2]: The ELF signal picked up by the large wire loop is first amplified, filtered and then given a DC offset of 2.5 V so that the signal swings between 0 and 5 V and does not go negative. The Arduino Uno uses an ATmega328 which already has a built-in A/D converter but with only 10-bit resolution. Kurt Diedrich did some research and ended up ordering a small PCB from Adafruit which has on-board an ADS1115 (from TI).

This particular A/D converter has four input channels and can measure signals with 16-bit resolution [3]. An I²C bus is used for control and to pass information between the chip and microcontroller. It wasn't long before Kurt was studying the Arduino code examples and software libraries supplied by Adafruit. The routines he needed to implement are not complex; after a little tinkering with the code examples the data logger firmware was finished. An endless loop in the

Figure 1. 16-bit data capture using an Arduino: The ADC module connects to the Elektor Extension shield via the Gnublin/EEC connector and the shield plugs into the Uno board.



main program repeatedly calls the function `ads.getLastConversionResults()` from the library, which returns the latest measurement value from the ADS1115 . This is then sent over the serial interface from the Arduino to the PC. There is no delay or timer control implemented in the firmware, the measurement frequency is only governed by the A/D converter's sample rate. New values are sent at a rate of about 112 Hz which Kurt was able to deal with in his PC software.

The concept

The prototype worked really well but the finished set up looked a bit untidy with all those flying leads between the ADC module and Arduino board. Labs set about making improvements; the main decision was to not make the design into a dedicated ADS1115 Arduino shield but instead opt for a more flexible approach that would allow the board to easily interface to a number of different systems [4].

The ADS1115 chip should derive its power from the attached controller board and be controlled via an I²C bus. A good choice for the interface would be the 14-way pin header used by the EEC/Gnublin boards. The ADC module can then be connected via a length of flatcable to the Elektor-Linux board [5], the Xmega-Webserver board [6] or the Elektor Extension Shield [7] (**Figure 1**). Screw clamp terminal blocks have been used for connection of the four analog input voltages. The chip cannot measure any voltage level which goes negative below 0 V so to measure alternating voltages it is possible to select an offset of around 1 V at the first analog input.

Voltages

The ADC board circuit diagram is shown in **Figure 2**. The Gnublin/EEC connector used by the board specifies a 3.3 V supply voltage but the ADC chip can operate with a supply range between 2.0 and 5.5 V.

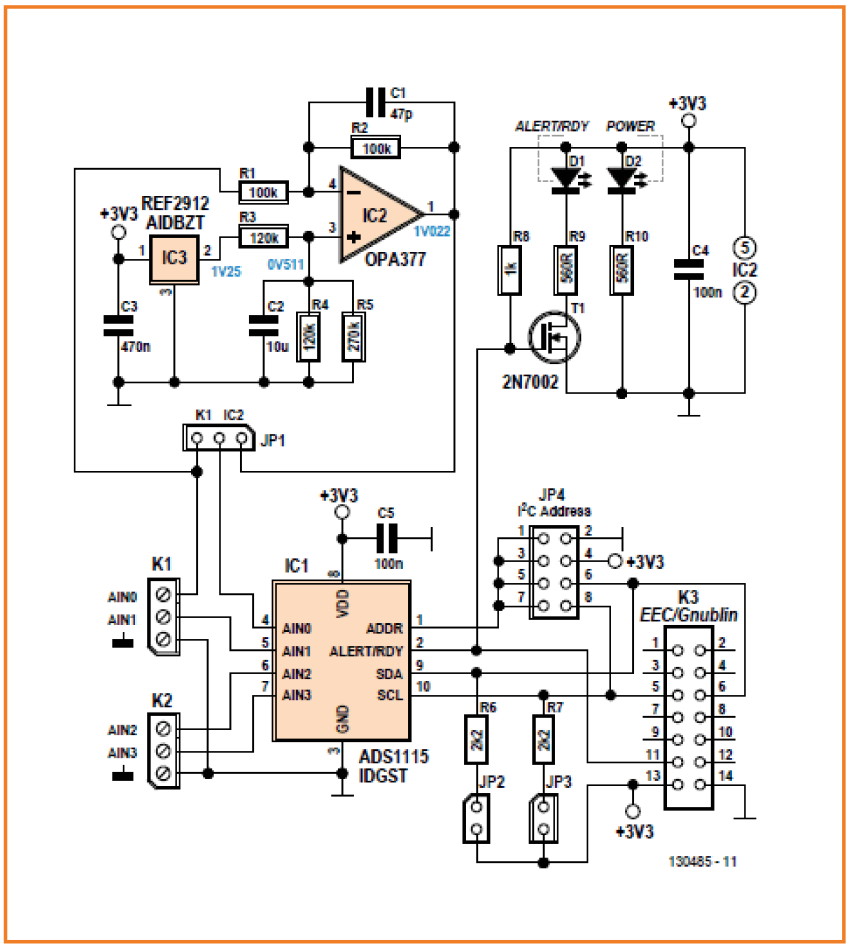


Figure 2. Schematic of the data logger ADC card.

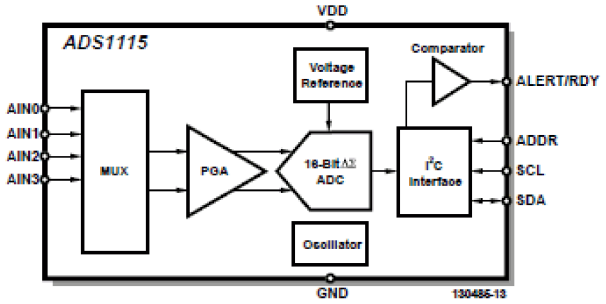


Figure 3.
The ADS1115 has a few built-in niceties such as an integrated comparator.

The data sheet [3] gives details of how the chip can be controlled and configured using I2C commands. It can be configured to measure:

- The voltage level on AIN0, AIN1, AIN2 and AIN3 relative to ground (i.e. four 'single-ended' inputs).
- The voltage level on AIN0 relative to AIN1 and the voltage level on AIN2 relative to AIN3 (two differential inputs).
- The voltage level on AIN0 relative to AIN3 and from AIN1 relative to AIN3 (two differential inputs with a common node).

The chip outputs a digital value in the range from –32768 to 32767; negative values indicate a negative differential voltage (negative values are expressed in twos-complement format). In single-ended mode the voltage will always be positive so that the output values will be in the range 0 to 32767 which corresponds to a 15-bit resolution. Using a workaround suggested by Ton Giesberts in our lab and described in more detail on the .Labs Website [8] it allows measurement of negative-going signals while producing a 16-bit value: Apply a fixed DC voltage, say 1 V on input AIN3. Now an input voltage of 0 to 1 V on the inputs AIN0 and AIN1 produces a negative output value when the corresponding differential mode is selected.

In addition to an internal voltage reference the ADC also has a built-in programmable amplifier which allows you to setup the full scale voltage range that the ADC will measure. In our case, with a 3.3 V operating voltage, the ranges ±2.048 V, ±1.024 V, ±512 mV and ±256 mV are of interest. With this board it is important to ensure the measured input voltage does not go above 3.6 V or below –0.3 V. The chip uses integrated ESD diodes on the inputs which provide some degree of protection but the data sheet recommends fitting external zener diodes and series resistors to give better protection.

Jumper JP1 can be fitted to provide a 1.022-V offset. The voltage level is made up of the 1.25-V reference voltage level produced by IC3 and the standard op-amp IC2. When the offset is applied the input voltage range is ±2.048 V. An AC input signal swinging between the values of around –1 V and +1 V will be translated into a voltage at the ADC input AIN0 swinging between 0 to 2 V with 15-bit resolution. In the software be aware that higher input voltages produce lower output values (inversion).

Another possibility is to add the offset to AIN0 and use the tip described above. With the measured voltage applied to AIN1 or AIN3, the corresponding differential mode configured and a full-scale range of ±1.024 V set. This method makes it possible to measure a voltage in the range from 0 to 2 V with a 16 bit resolution.

Timing

The ADC operates using the Delta-Sigma principle, which gives good resolution and accuracy. This method of A-D conversion however is not the fastest. A rate of up to 860 samples/s is possible. The sample rate is configurable and has a default value of 128 samples/s. Like many other ADCs the ADS1115 can work in single-shot or continuous mode. In single-shot mode it is necessary to issue a command to the chip to tell it to measure the input voltage. Once the measurement is complete and a valid digital value is available the chip will set a bit in one of its internal registers. The external microcontroller monitors the state of this bit and reads the new value from an

internal register when it becomes available. It can then go on to request a new measurement. In continuous mode the chip continually measures the input voltage and produces digital values without the need for any external intervention.
In all our programs including the ELF receiver application we use single-shot mode in a continuous loop. After each new reading (and before issuing the next sample command) the value is processed in the microcontroller (e.g. to show on the display or send over the serial interface). Using the default sample rate of 128 samples/s produces a data rate of about 100 to 120 Hz which should be sufficient for the majority of data logger applications.

The comparator

The Conversion-Ready-Pin (IC1 pin 2) can be used to indicate when an A-D conversion has been completed. The ADS1115 can also be configured to act as a comparator (**Figure 3**) and in this mode the pin acts as an alert output. Now using values stored in the Low Threshold and High Threshold registers you can allow a sort-of hysteresis or apply a voltage comparison window to the measured value so that an alert occurs only if the measured value is outside these stored values.
In our module this output is made visible via transistor T1 and a yellow LED. In addition this signal is routed to pin 11 of the Gnublin/EEC connector K3 where it is available for use by either the Linux- or Xmega-board but has no connection on the Elektor extension shield.

I²C

As we already mentioned communication with the ADC occurs over an I2C interface where it plays the role of a slave. Using just write operations you can, for example set up an internal configuration registers or tell the chip to start a measurement process. In order to read out a digital value we need to both write and read over the I²C bus. To be precise you need to write twice (which includes the slave address) and then read once. The chip's slave address can be set-up externally so it allows more than one ADS1115 chip to be connected to the same two bus wires (provided you use different addresses for each chip). The slave address is defined by how pin 1 is connected. There are four possible slave addresses available; pin 1 can be linked to either pin 3, 8, 9 or 10 using a jumper on the header pins of JP4 to define the address.

The two I2C signals are connected from the chip to pins 5 and 6 of the EEC/Gnublin connector. Jumpers JP2 and JP3 allow you to use the on-board pull-up resistors. The beauty of the I²C bus is that you can safely interface a slave device operating at 3.3 V with a controller such as an Arduino Uno running at 5 V.

The software library

When it comes to the control software you know that even if you opt for the default settings of the ADC you will still need to spend some time studying its data sheet. Controlling a microcontroller's I²C interface can also sometimes be a bit tricky. Don't worry; we have simple solutions that should get around these potential problems. The technical boss at the Elektor Labs Clemens Valens has written a C library that avoids the need to define which registers you need to address in the ADC and instead provides high level functions to control the chip. As a bonus he has also

Web Links

[1] www.elektor-labs.com
[2] www.elektor-labs.com/project/arduino-16-bit-low-frequency-datalogger-130485-i-140035-i.13703.html
[3] www.ti.com/lit/ds/symlink/ads1115.pdf
[4] www.elektor-magazine.com/130485
[5] www.elektor-magazine.com/130214
[6] www.elektor-magazine.com/120126
[7] www.elektor-magazine.com/140009
[8] www.elektor-labs.com/contribution/from-the-lab-using-differential-mode.14034.html
[9] www.elektor-magazine.com/120668

Figure 4.
To test the ADC module and the Data logger application connect a wire from the Arduino pin A3 to the ADC input at AIN0.

integrated a small I2C software library, it uses a bit-banging technique to generate the I²C signals. This means that they are also applicable to controllers that don't have a built-in hardware I²C interface. The complete implementation can effectively be used with any platform for which there is a C compiler. The files, together with a brief description are available to download from the web page related to this article [4].

First off, bind the ads1x1x.c/.h and soft_i2c.c/.h files to your own software project. Make sure you refer to them with an include statement in your code. In order to get it on your own microcontroller, it is necessary to implement the following five functions (which shouldn't be too difficult):

```
void soft_i2c_scl_write(uint8_t value)     // write high or low state to the SCL pin
void soft_i2c_sda_write(uint8_t value)     // write high or low state to the SDA pin
uint8_t soft_i2c_sda_read(void)            // Read the SDA pin state
void soft_i2c_sda_mode(uint8_t value)      // configure SDA pin as O/P (value=1) or I/P (value=0)
void soft_i2c_delay(void)                  // 8 µs delay
```

Five further functions bind the ADC library (from Clemens) with the I²C software library, this code can just be dropped 1:1 into the project.

As a bonus the functions

```
ADS1x1x_start_conversion(&my_adc);
```

and

```
uint16 Result = ADS1x1x_read(&my_adc);
```

are available.

`my_adc` is a structure, defined at the start of the program (`ADS1x1x_config_t my_adc;`) which is filled via the function `ADS1x1x_init(…)` with the required configuration parameters. All following commands will be given a pointer to this configuration structure. More structures can be defined so that the library can allow more chips to be addressed on the same bus and independently configured.

Using the following command, for example indicates that the slave chip on the bus with its address pin connected to ground will be addressed. The input AIN0 will be used in single-ended mode, the measuring range is –2.048 V to 2.048 V.

```
ADS1x1x_init(&my_adc, ADS1115, ADS1x1x_
I2C_ADDRESS_ADDR_TO_GND, MUX_SINGLE_0,
PGA_2048);
```

The second parameter indicates that an ADS1115 chip is addressed, the library is also usable for the other ADS111x devices.

The Data Logger software

To get a good overview of the design it is a worthwhile exercise to study the source code of the data logger Arduino sketches, which uses the libraries referred to above [4]. In the next issue you will find more demo programs and interesting applications which make use of the Embedded Firmware Library [9].

Component List

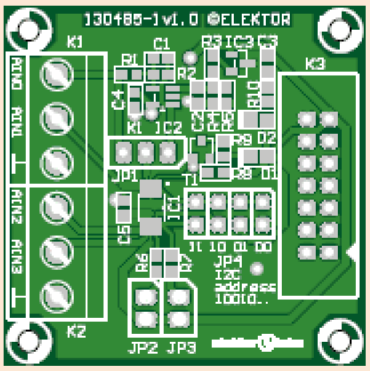
Resistors (0.1W, 0603)
R1,R2 = 100kΩ 1%
R3,R4 = 120kΩ 1%
R5 = 270kΩ 1%
R6,R7 = 2.2kΩ 1%
R8 = 1kΩ 5%
R9,R10 = 560Ω 1%

Capacitors (0603)
C1 = 47pF 5%, 50V, C0G/NP0
C2 = 10µF 20%, 6.3V, X5R
C3 = 470nF 10%, 10V, X5R
C4,C5 = 100nF, 10% 16V, X7R

Semiconductors
D1 = LED, yellow, 0805
D2 = LED, green, 0805
T1 = 2N7002 (SMD SOT23)

IC1 = ADS1115IDGST
IC2 = OPA377AIDBVT
IC3 = REF2912AIDBZT

Miscellaneous
K1,K2 = 3-way PCB screw terminal block, 0.2'' pitch
K3 = 14-pin (2x7) pinheader, 0.1'' pitch
JP1 = 3-pin pinheader, 0.1'' pitch, with jumper
JP2,JP3 = 2- pin pinheader, 0.1'' pitch, with jumper
JP4 = 8-pin (2x4) pinheader, 0.1'' pitch, with jumper
PCB 130485-1



Our Arduino sketch performs the same software function as Kurt Diedrich's original project: An endless loop repeatedly tells the ADC chip to measure and digitize the voltage at input AIN0 in single-ended and single-shot mode. The measurement range extends from −2.048 V to 2.048 V, while in Single-ended mode only 0 to 2048 mV can be measured and with only 15-bit resolution. The decimal output values (0 to 32767) after conversion are sent out as ASCII characters over the serial interface, followed by a <CR> and <LF> character.

For testing we used an Arduino Uno fitted with the Elektor extension shield described in the last issue [7] connected to the ADC board via a 14-way ribbon cable. The I2C address of the ADC board is configured as 'ground' i.e. the jumper is placed in the first position of JP4 nearest the Gnublin/EEC connector. Initially we will not use any offset correction (JP1 jumper fitted nearest the terminal blocks).
Now to generate the voltage to measure we connect a flying lead from the Arduino pin A3 (you can connect it at the shield box header connector as shown in **Figure 4**) to the AIN0 input of the ADC board. The voltage on A3 can now be adjusted by twiddling the pot on the shield; this gives us a rudimentary test set-up to check operation of the ADC board. The voltage at A3 can be turned up to 5 V but it's important that the input voltage level does not exceed 3.6 V so before connection ensure that the pot is turned all the way down so that the voltage is at 0 V then slowly increase it.

It is probably easiest to upload the sketch using the Arduino bootloader, that way you only need a USB cable between the Arduino and PC. Once the firmware has been flashed the measured values can be seen on the serial monitor display in the Arduino development environment or via a simple terminal emulator program (data rate set to 115,200 Baud and the corresponding COM port selected).
To achieve the highest measurement accuracy it is better to power the Arduino from a bench power supply rather than from a USB port. It will provide much better common-mode noise performance.

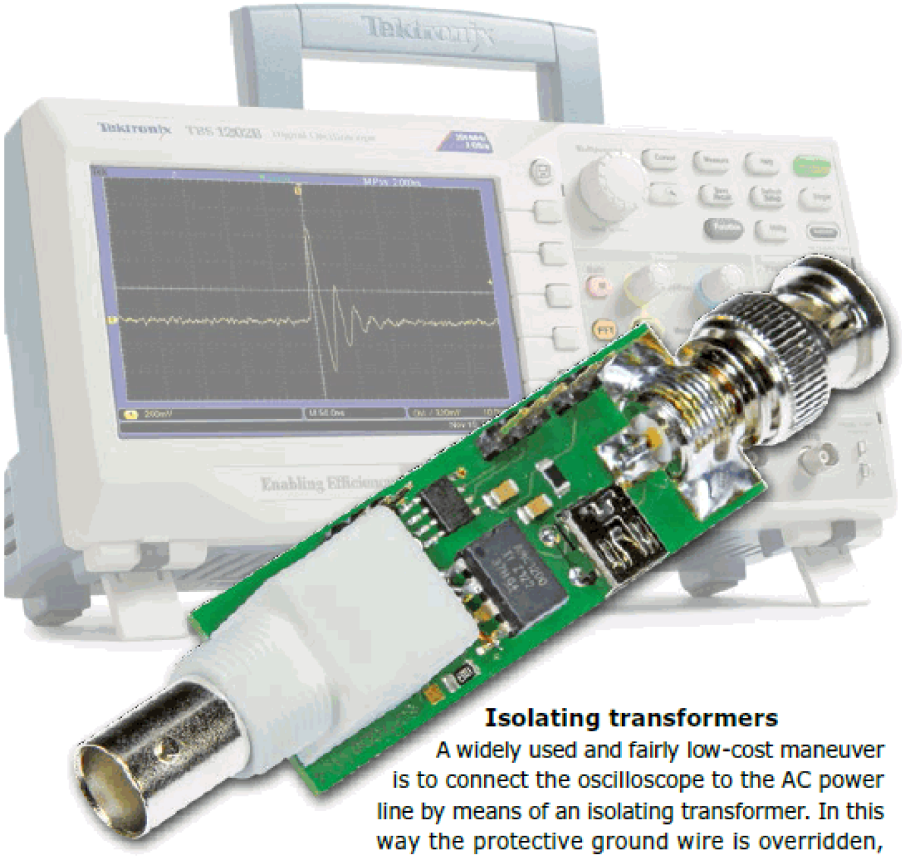
Some spooky signals

If you like you can now try out the recorder software that Kurt Diedrich developed for a his ELF reception experiments. Now when you turn the pot the values are recorded over time in software. You can also switch in the 1022 mV offset and see how the measured values change. Voltage levels in the range of approximately −1 V to +1 V can be measured with 15-bit resolution, this set up can be put to good use investigating the presence of ELF signals in your location. If this sounds interesting have a look at the 'ELF Receiver' project described in this issue you may be in for some surprises!

(130485)

Isolated Oscilloscope Probe

Petite and practical



By **Erik Lins** (Germany); erik.lins@chip45.com

However much you might wish for an oscilloscope with electrically isolated inputs, it's hard to justify the cost for personal projects. Even differential probes, which (within certain limits) enable voltages to be measured without reference to ground, often cost the private user more than a complete scope does. So what can you do when either safety considerations or the nature of the task in hand require the use of isolated connections to your oscilloscope?

Isolating transformers

A widely used and fairly low-cost maneuver is to connect the oscilloscope to the AC power line by means of an isolating transformer. In this way the protective ground wire is overridden, eliminating any danger of live circuit elements becoming grounded when the probe is attached. Any risk of a short circuit and possible destruction of the circuitry is thus averted. Conversely a clear hazard remains, in that the probe leads might become energized at high potential (AC line voltage for instance), putting parts of the oscilloscope equally at risk. In particular the BNC connectors of channels not in use at the time could be rendered live (and these are not protected against accidental touch).

When multiple probes are connected, these too could become live, so ideally they need to be placed where there is no risk of accidental contact with them. In the worst case these probes might be connected to other part of the circuitry under

Features

- Electrical separation of 1 analog signal and 2 digital signals
- Max. input voltage ±250 mV (or ±2.5 V/±25 V, determined by jumper)
- Amplification factor between input and output: 8 x (without voltage divider on input)
- Signal bandwidth, analog input: 60 kHz
- Signal bandwidth, digital inputs: 1 Mbps
- Power supply via separate AC adapter or Mini USB connector
- Power requirements: 5 V; 110 mA

examination and cause the isolating transformer to become short-circuited.
For these reasons the isolating transformer method is reliable only for making electrically isolated measurements on one single channel, at low voltages. When two channels are involved, with most scopes you can, in principle, make differential measurements (e.g. channel 1 minus channel 2), but the input wiring of the channels imposes limitations on matters such as common mode rejection range. Accordingly the isolating transformer method should not be first choice for an experimenter taking measurements on high voltages.

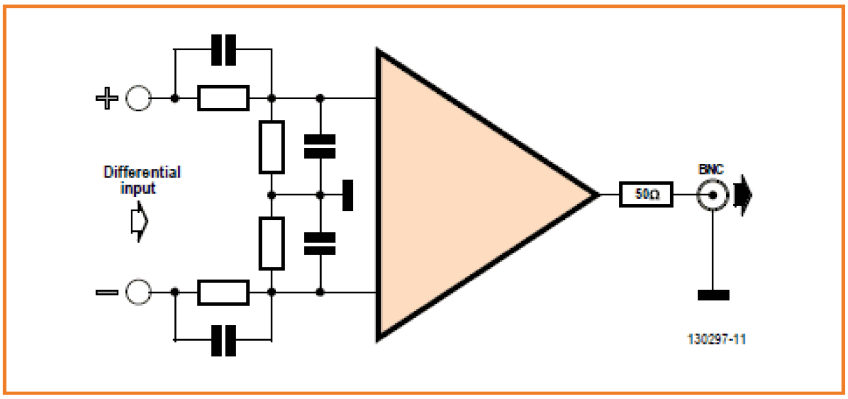


Figure 1.
Block diagram of a differential probe. The input circuitry consists of the voltage divider (high impedance series resistor) and comparatively low resistance resistor in parallel at each input.

Differential probes

Moving on, differential probes are the next more expensive method for making measurements without reference to ground. The illustration in **Figure 1** shows the schematic circuit of a differential probe. The input wiring consists of a voltage divider (high-value series resistance) on each of the Positive and Negative inputs, together with a comparatively low-value resistor in parallel with the input of the opamp. The high-impedance connection to the inputs ensures that even at high offset voltages only a small current flows through the series resistors. Since the offset is generally the same for both inputs (e.g. high-side shunt measurements), the differential voltage does not upset the opamp.
The limiting factor here is the common-mode suppression of the opamp, since at larger offset voltages the common mode rejection range is exceeded sooner or later. In every case the limit is the supply voltage of the opamp. In practice these are often provided with a voltage of ±4.5 V supplied from a 9-V 6F22 (PP3-size) battery. Accordingly, with a voltage divider of 10:1, the maximum voltage you should connect to the inputs of a probe is ±45 V. If you raise the division factor to 100:1, theoretically you could go up to ±450 V and in this way use a shunt to take measurements across a 230 VAC or 115 VAC circuit. That said, you must operate only in voltage ranges for which the components used in the input circuitry are rated for adequate dielectric strength. In other words, not every differential probe is suitable. In addition, the greater the division factor for the input voltage, the more you correspondingly reduce the signal being measured. This (if you are using shunts) is already tiny, since shunts are configured with as low resistance as possible to prevent power dissipation. For example, a useful signal of ±250 mV with an offset of +40 V produces, for a division ratio of 10:1, a signal of ±25 mV with 4 V offset. If we set the oscilloscope for 1 V/Div, to center the 4 V at the middle of the display, the wanted signal amounts to just 1/40 Div. Even if you set the scope to show 0.1 V/Div, the signal we are trying to display amounts to just 1/4 Div, which corresponds to a resolution of only 3 bits (assuming 8-bit vertical resolution). On top of this, to see the entire signal on the display you would need to displace the vertical position of the oscilloscope by -40 V, which is barely feasible even on expensive scopes.

Hopeless task?

Given that the restricted common mode rejection range and the undesired reduction of the wanted signal produce contrary demands on the input circuit design, there is always a compromise when using differential probes. However, the use of electrically isolated probes offers an elegant solution here. This approach allows unrestricted connection to high voltages, without any high offset voltage arising on the inputs of the internal opamps. Moreover, an electrically isolated probe has advantages even when high offset voltages are not involved, such as when ground loops must be avoided.

Selecting suitable components

The Internet is brimming with homebrew circuits for isolated probes and discussions about the hookups employed. Nevertheless it soon becomes apparent that most of these solutions rely on using differential probes that do not provide genuine electrical isolation. To keep our circuit simple, we too settled against a true differential probe and opted for a circuit using a simple isolation

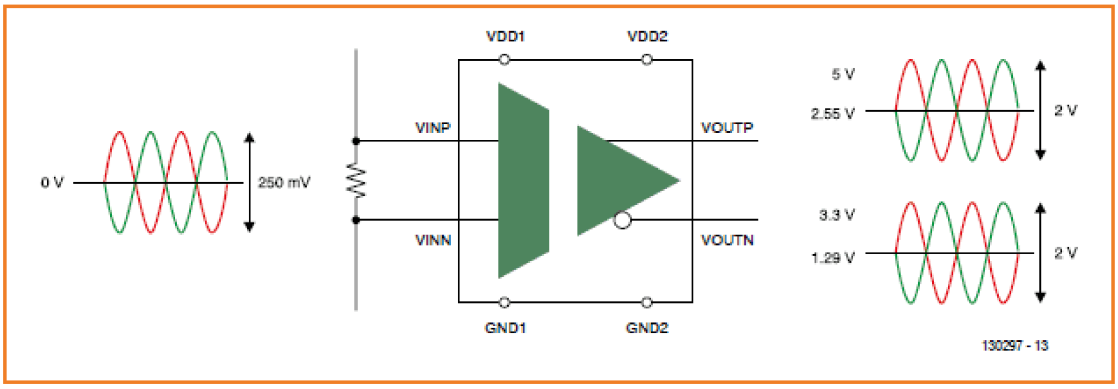


Figure 2.
The AMC1200 is a "fully differential isolation amplifier" capable of handling input voltages up to ±250 mV.

amplifier for a single probe, as this seemed to represent the commonest need in semi-professional circles.

For its functioning an electrically isolated opamp requires two separate supply voltages that are each isolated electrically from one another. The supply for the output side can share the same ground potential as the oscilloscope, whereas the input side must not share any ground reference with the scope. The simplest solution is to use a pair of 9 V 'block' batteries to produce the ±4.5 V supply voltage (one for the input side of the opamp and one for the output). The physical size of two 9 V batteries (each H: 48.5, L: 26.5, W: 17.5 mm) plus the constant need to have two fresh batteries on hand are admittedly distinct disadvantages.

An electrically isolated DC/DC converter is another possibility for powering the input of the opamp, making one additional single supply voltage necessary for the output side. The typically low current demand of an opamp permits the use of a small DC/DC converter, requiring few external components. The secondary voltage must naturally go well with the electrically isolated opamp employed.

The search for an appropriate opamp didn't take long and we rapidly opted for the well-known ISO series from Burr Brown (now Texas Instruments), specifically the ISO124 [1]. Its isolation rating is 1500 V with an amplification factor of 1 (unity gain) and a supply range from ±4.5 V up to ±18 V. The signal bandwidth for the ISO124 is typically 50 kHz. This is of course rather small, even when considered against the 10 to 100 MHz bandwidth of a low-cost oscilloscope. Nevertheless the analog voltages encountered in the semi-professional range would typically not exceed this bandwidth.

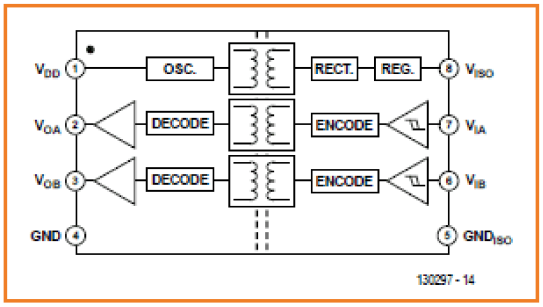


Figure 3.
Block diagram of the AduM5242, a two-channel isolator with integrated DC/DC converter.

However, to keep costs down we opted for the AMC1200 [2] from the same manufacturer. Once again we're talking about an electrically isolated opamp (**Figure 2**), which is configured primarily for taking high-side shunt measurements. It operates from a simple supply voltage of 5 V on each side and has an input voltage range of ±250 mV plus a fixed gain factor of 8. So with ±250 mV at the input we have ±2 V on the output. The signal bandwidth is 60 kHz, which in our opinion is satisfactory for this application. Current consumption is agreeably low at typically 5 mA, enabling a small DC/DC converter from the iCoupler range made by Analog Devices to be used along with a straightforward 5 V supply. What we have here is an electrically isolated coupler predominantly for digital signals, e.g. such as SPI or I2C interfaces. Some of the components in this range, however, have in addition to the couplers for the digital signals a low-power 5 V/5 V DC/DC converter on board, providing what AD calls "isoPower". Primarily this is for serving the coupler's own supply needs but it has a small power reserve that is adequate for our AMC1200. The ADuM5242 [3] we selected comes in SOIC-8 package format and is therefore easy to solder by hand. In addition to the DC/DC converter there are also two digital couplers on

board (**Figure 3**), enabling our circuit not only to measure the electrically isolated analog voltage but also to handle two digital signals, again in electrical isolation.
Two pin-compatible variants exist also, the ADuM5240 and 5241, offering two independent isolation channels in a variety of channel configurations. Configured appropriately, these can be used for electrically isolated control of digital signals in the measurement circuitry. One disadvantage of the iCoupler chip and its integrated DC/DC converter is the rather restrained efficiency factor of the converter, which is less than 20 %.

We now have an all-5 V solution; 5 V supplies are widely available and should be easy to find virtually everywhere. If nothing else, many modern oscilloscopes possess a connector for USB memory sticks, through which our circuit could also be powered.

For sake of completeness it should also be mentioned that significantly more powerful isolation amplifiers are also available, such as the AD216 from Analog Devices [4]. This offers double the bandwidth (120 kHz) and even contains an integrated DC/DC converter for the electrically isolated supply for the input side. It does, however, require a bipolar power supply of ±15 V and costs six times the price of the combination of ADuM5242 and AMC1200.

The circuit

The circuit in **Figure 4** is comparatively simple and consists essentially of the components already described. The ADuM5242 (U1) generates the electrically isolated voltages 5 V (VCCiso) and 0 V (GNDiso) for the input side. The two electrically isolated digital channels are taken to a simple three-pin connector strip CON5. The supply for the AMC1200 (U2) is fed via a small choke, which smoothes the output voltage of the ADuM5242 a little more.
The negative opamp input is connected to VCCiso/2 via a 1:1 voltage divider (R5/R6), so that the voltage on the positive input at this point can swing ±250 mV at full range. To extend the measurement range to ±2.5 V or alternatively ±25 V we employ the voltage dividers R2/R3 and R2/R4, activated using jumpers J1 and J2 respectively. The jumper settings are as follows:

1:1	J1: closed	J2: open
10:1	J1: open	J2: 1-2
100:1	J1: open	J2: 2-3

As the voltage dividers have very high resistance (in order to load or degrade the signal source as little as possible) as opposed to the AMC1200, which with 28 kΩ has relatively low differential input resistance, we employ the simple opamp U3 in the signal path as an impedance converter. The oscilloscope probe is connected via CON6

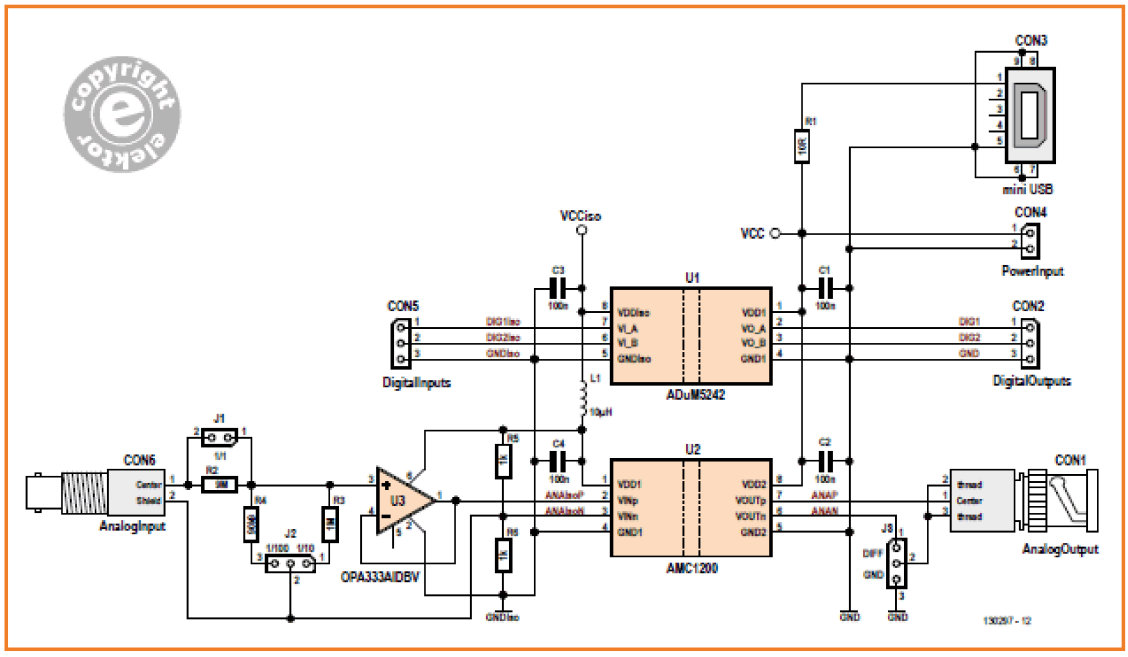


Figure 4.
The circuit is relatively simple and consists essentially of an ADuM5242 (U1) and an AMC1200.

Component List

Resistors
(SMD 1206)
R1 = 10Ω
R2 = 9MΩ
R3 = 1MΩ
R4 = 100 kΩ
R5,R6 = 1kΩ

Capacitors
C1–C4 = 100nF (SMD 1206)

Inductors
L1 = 10µH (SMD 1206)

Semiconductors
U1 = AduM5242 (SOIC-8)
U2 = AMC1200 (SOP-8)
U3 = OPA333AIDBV (SOT-23)

Miscellaneous
CON1 = BNC plug, panel mount
CON2,CON5 = 3-pin pinheader, 0.1" pitch
CON3 = mini-USB receptacle
CON4 = 2-pin pinheader, 0.1" pitch
CON6 = BNC socket, panel mount
PCB # 130297-1 [5]

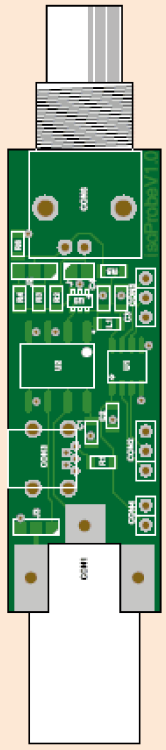


Figure 5.
The circuitry goes together rapidly on this small printed circuit board.

to the input of the voltage divider and the negative input of the AMC1200. At this point the shield terminal of the probe certainly is not at GNDiso potential but this is not problematic, for one thing because the circuit is "suspended in mid-air" and entirely potential-free thanks to the electrical isolation. The other reason is to enable the input voltage to then swing versus the ground connection according to the setting of the voltage divider by respectively ±250 mV, ±2.5 V or ±25 V.
Problems arise only when the digital channels are in use and for this GNDiso (pin 3 of CON5) needs to be linked to the ground of the measurement circuit. Doing this also takes the negative input of the AMC1200 down to ground, with the measurement range becoming then only a unipolar +250 mV (+2.5 V/+25 V), relative to the shield terminal of the probe. Nevertheless this is no more than only a minor limitation, since the circuit is designed primarily for making analog measurements and the two digital channels are purely a bonus gift from the ADuM5242.
Incidentally, instead of using the voltage divider specified, our circuit can also operate with a 10:1 or 100:1 probe to increase the measurement range (J1: closed, J2: open).

The connections to the power supply and probe input from the oscilloscope CON1 are located on the output side of the circuit. The best possible performance is achieved when the circuit is powered, via CON4, from an electrically isolated lab power supply unit. The supply voltage is then fully isolated from the oscilloscope and the scope input can be hooked up to the differential outputs of the AMC1200 (jumper J3: 1-2). The signal then swings (at full range) by ±2 V either side of oscilloscope ground. The gain of 8 in the AMC1200 produces a differential aggregate transfer factor, from input to output, of 1:8, 10:8 and 100:8, which correspondingly needs to be taken into consideration when taking readings of the oscilloscope voltage. If your scope provides fine adjustment of the vertical graduation, you can alter this to, say, 125 mV/Div and then take readings with the circuit set to 1:8, as if you had set the scope to 1 V/Div.

The circuit also includes a 5-pole Mini USB-B connector for obtaining power from, say, the USB memory stick connector on an oscilloscope. In this situation the ground connection of the probe input CON1 will be at GND potential at the oscilloscope end and jumper J3 must then be set at position 2-3. If this was not done, negative AMC1200 output would be short-circuited to GND. In this configuration the output signal on the oscilloscope has a fixed offset of around VCC/2 (2.55 V, see AMC1200 data sheet) and the centre point amounts to only half of this, since we are using only one out of the two differential output signals currently. In this situation the transfer factors mentioned above are halved correspondingly. Taking the power supply from a USB connection is really an emergency expedient, as the purity of the voltage on a USB bus is not adequate for supplying analog circuitry and will be affected every now and again by interference arising from the inner digital regions of the scope.

Construction

The circuit uses only a few components and since these are predominantly large SMD devices, the PCB in **Figure 5** can be assembled rapidly. The PCB layout can be downloaded free from the project page [5].
The ADuM5242 comes in a SOIC-8 package and the AMC1200 in an even larger SOP-8 package, corresponding in essence to a DIP-8 package

equipped with angled pins for SMD mounting. All resistors and capacitors have the 1206 form factor, enabling them to be soldered without difficulty. The impedance converter U3 differs in having a small SOT-23 package and for this reason should be soldered into place with care as the first component to be fitted, followed by the ADuM5242 and the AMC1200. After this come the SMD resistors and capacitors and the wound components. Since BNC plugs for 90° mounting on printed circuit boards are extremely uncommon, we have provided the PCB with a suitable cut-out to enable a normal panel-mounting BNC connector to be fitted. To solder it in place you will need a soldering iron with adequate wattage to bring the metal thread of the connector rapidly to the melting point of the solder, without degrading the plastic insulation inside. The unit is remarkably compact and can be inserted conveniently between the oscilloscope and probe.

Commissioning

As a result of the low efficiency factor of the ADuM's DC/DC converter (less than 20 %), the current consumption of the complete circuit is around 110 mA. This figure should be checked after constructing the circuit and connecting it to a 5 V power supply. If you have one, it's advantageous to use a laboratory power supply with presettable current limiting. From preference, commissioning should not be carried out taking the supply from a USB connector. If the current drawn is significantly greater than the value quoted, checks should be made for possible soldering errors, short circuits or components inserted back to front. After this you should measure the supply voltage VCCiso on the secondary side, which is typically 5.2 V. If both of these values are correct, the circuit can be connected between oscilloscope and probe; you can then check out its correct functioning with a signal source and oscilloscope.

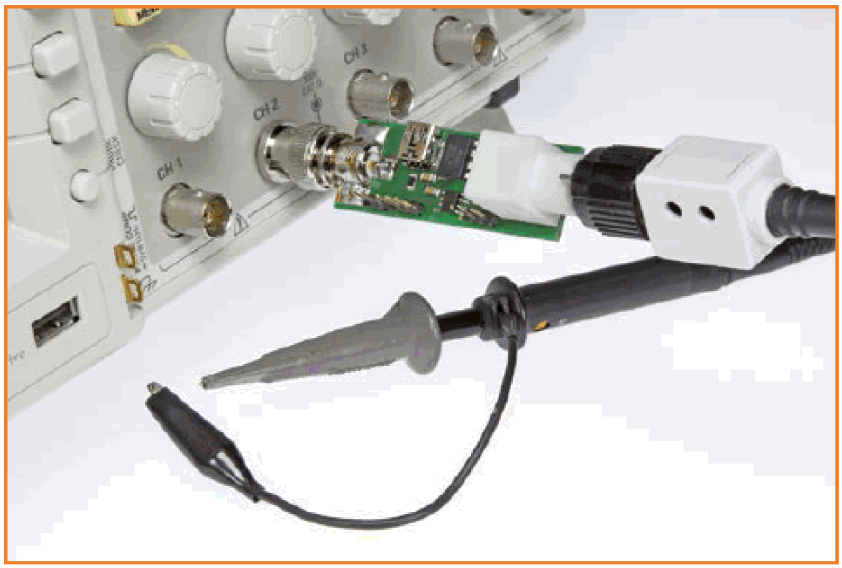


Figure 6.
The circuit put into practical use. The PCB of the prototype seen in this photo differs from the board shown in Figure 5.

Verdict

Using modern, low-cost components it's easy to construct a simple isolation amplifier for oscilloscope probes. The restricted analog bandwidth does admittedly restrict the potential applications, meaning that it's not feasible to measure, for example, the signal quality of differential high-speed signals (such as LVDS, USB or Ethernet). On the other hand, the true electrical isolation provided can occasionally be advantageous for connecting expensive differential probes, if the signal/offset relationship is of importance or when avoiding ground loops is a priority. The modest cost means you can also make several units of this circuit and equip every channel of your scope with electrical isolation. This isolation is provided not merely for the device under test, as could also be achieved with an isolating transformer; doing it this way, every channel is isolated from each other. Consequently you can take potential-free measurements at different points in a circuit with just one single oscilloscope.

(130297)

Web Links

[1] www.ti.com/product/iso124
[2] www.ti.com/product/amc1200
[3] www.analog.com/en/interface-isolation/digital-isolators/adum5242/products/product.html
[4] www.analog.com/en/specialty-amplifiers/isolation-amplifiers/ad215/products/product.html
[5] www.elektor-magazine.com/130297

DesignSpark Tips & Tricks

Day #13: Component placement

By **Neil Gruending**
(Canada)

Today let's look at DesignSpark's component placement tools.

In the previous installments of this series we've spent quite a bit of time looking DesignSpark's schematic features. Now, let's look at some of DesignSpark's PCB features, starting with automatic component placement. Then we'll look at some manual placement techniques.

Automatically placing PCB components

Let's try out DesignSpark's automatic component placement tool on the LED driver board we designed in an earlier installment. Go into the Tools → Auto Place Components → All Components and you will see the Autoplace Components window shown in **Figure 1**. This is where you configure the Autoplace tool before running it.

I set the minimum component spacing to 0.25 mm and the placement grid to 0.25 mm and got the result in **Figure 2**. As you can see, all of the components are spaced properly and DesignSpark did its best to organize the parts on the board.

But what if you didn't want all of the components automatically placed on the PCB? If you select "All Unplaced" from the "Auto Place Components" menu DesignSpark will take any parts that aren't on the board and put them on the board for you. You can also select one or more components and choose "Selected Components" to place just those components. Both of these commands use the settings that were last used for the "All Components" command.

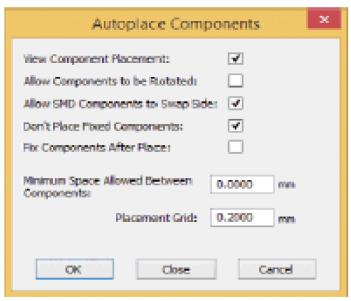


Figure 1.
Autoplace window.

Sometimes you might want to place almost all the components in a design but leave some fixed in their current location. For example, you could have some mounting hole components on the board that you want to keep put while placing all the other components on the bard. One solution is to lock or fix the mounting hole components by right clicking on them and choosing "Fix Item" and the make sure that "Don't place fixed components" is checked in the placement settings. DesignSpark will also check and skip any components that have any traces connected to their pads while placing components. That way you can be sure that no components will be disconnected from their nets. However, DesignSpark will not check if the location where it wants to place a component is free of traces so sometimes it can short out nets like in **Figure 3** where Q2 is accidentally connected to the trace between Q1 and Q4. For a larger design I would use the design rule check (DRC) and these kinds of errors would show up as track to pad clearance errors.

Manually placing components

Want to manually place the components instead? Clicking and moving parts around in DesignSpark PCB is pretty straightforward but it's difficult to know how far apart the components are from each other. The automatic placement tool can do this measurement itself but when you do the placement by hand you're responsible for the component clearances yourself. The most straightforward way to do this is to use a placement grid like 0.25 mm and then make sure that there's enough full grid spaces between each component for your desired spacing. For example you would place two components as close as possible without overlapping them and then move one of them enough grid space horizontally or vertically as needed to get the required space. This works well for small boards like out LED driver example but quickly gets tedious if different component spacing is needed or the board has a large number of components.

The other option is to use placement courtyards or placement areas in your library components. A

placement courtyard is just a square box drawn on a mechanical layer that includes the component outline and any component tolerances. Technically they aren't supposed to include manufacturing and assembly requirements but I usually add a bit of room to make sure my components are at least 0.25 mm from each other. When you place the components on the board, you can space them out however you want as long as you don't overlap the placement courtyards. Let's see how to do this in DesignSpark.

When you use the footprint wizard when making a new component, it will ask you if you want a placement outline as the last step which is the same thing as a placement courtyard. You just have to tell it how far you want it from your pads and then it will be automatically added to the footprint. I usually use the top assembly layer for the courtyard so that it's easy to use it as the component outline in an assembly drawing later. You can also manually add the placement courtyard to an existing component like in **Figure 4**. But before you update the footprint in the PCB you need to add the assembly layers first. Open the "Design Technology" window in the Settings menu and click on the "Layer Types" tab. Click on the Add button and give it a name, I used Assembly. The usage should be Non-Electrical and don't include anything by unchecking all of the checkboxes in the include area. In the Settings area, check the Placement Shapes box and finally the Ok button.

Now you can add the top and bottom assembly layers to the PCB by clicking on the Layers tab. Click the Add button and name the layer Top Assembly. Then change the type to Assembly and choose a color. Click on the Ok button and repeat the steps to add a bottom assembly layer, but don't forget to change the side to Bottom.

Once that's all done our example looks like **Figure 5** with a little bit of rearranging. All of the placement courtyards make it easy to see when the components are spaced correctly when none of the courtyards are overlapping or touching.

Conclusion

Today (or is it this month ☺) we looked at various ways to place components on your PCB. I prefer the placement courtyard method but the automatic placement tool is great for quickly evaluating a board to see how well the component fit. Next time we'll look at DesignSpark's routing features.

(140046)

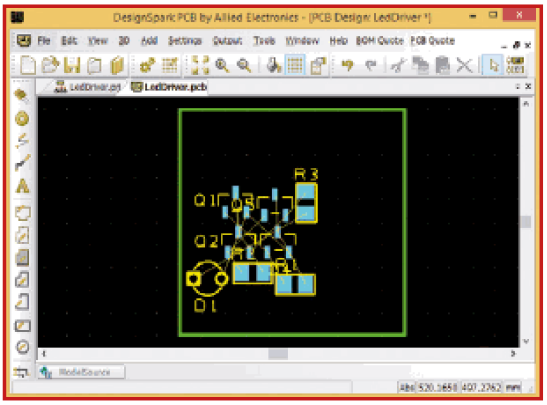


Figure 2.
Automatic placement example.

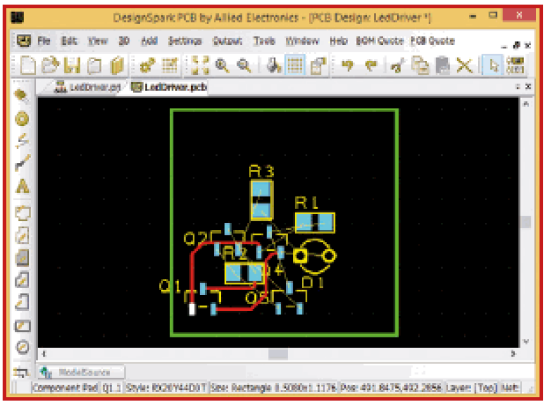


Figure 3.
Placement error.

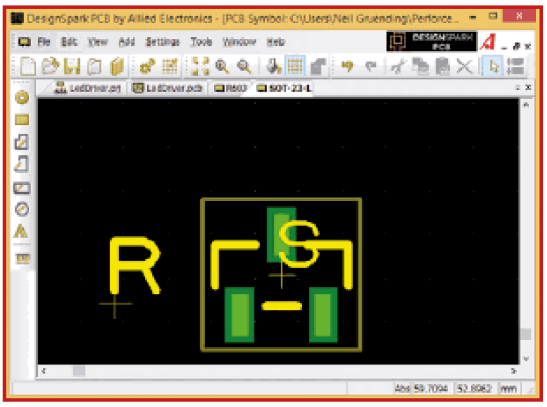


Figure 4.
SOT23 placement courtyard.

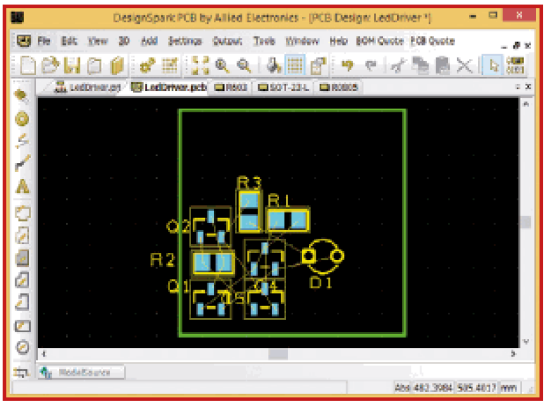


Figure 5.
Board with placement courtyards.

Peltier Modules

Weird Component #7

Have you ever wondered how one of those small, portable coolers keep things cold without any moving parts? Well wonder no more—they use small, solid state cooling systems based on Peltier modules. Let's take a look at how the modules work.

By **Neil Gruending**
(Canada)

A junction of dissimilar metals creates a thermocouple which will generate a small voltage that varies with temperature. However, if you apply a voltage to a thermocouple it will create a temperature differential—the phenomenon is known as the Peltier effect. When you combine many thermocouple connections in the form of heavily doped P and N silicon together you actually create a heat pump commonly known as a Peltier or Thermoelectric module. **Figure 1** shows how these devices are constructed.

Peltier modules aren't 100% efficient so they require power in order to transfer heat from one side of the device to the other. This means that when the hot-to-cold temperature differential is zero, the Peltier module only needs to dissipate the heat from its own operation. But as the temperature differential increases the module becomes thermally limited, which decreases the amount of heat it can transfer. In practice the typical upper limit is about 70°C. **Figure 2** shows what the transfer function looks like.

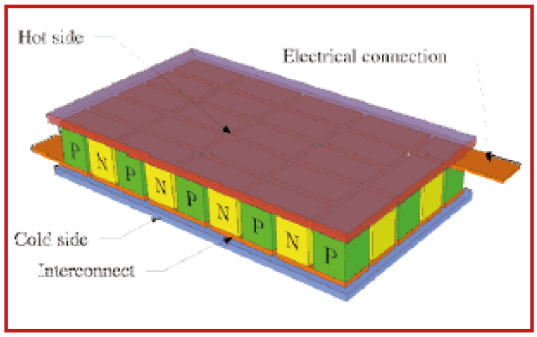


Figure 1.
Peltier module construction.
(creator: "michbich" under Creative Commons license)

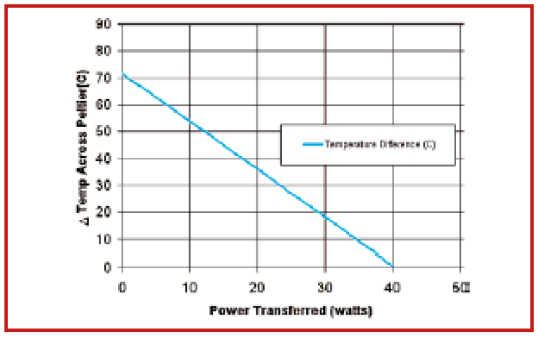


Figure 2.
Peltier performance curve.
(source: heatsinkguide.com)

The main advantage of Peltier modules is their reliability due to being all solid state hence not requiring any moving parts like other mechanical cooling methods. They are also available in a wide variety of sizes, and are easily controlled by varying their input voltage/current. It's also possible to switch between heating and cooling operation by reversing the current flow through the device. The downside is that Peltier modules are only about 25% as efficient as mechanical cooling which limits their practical cooling or heating capacity. Another issue is their limited heat transfer ability, especially over wide temperature ranges.

Peltier modules have another unique feature where they can be used as thermoelectric generators, converting heat into electricity. They aren't terribly efficient but they can generate a reasonable amount of power depending on the configuration. The big challenge is dealing with their high output impedance and their limited thermal conductivity.

Peltier cooling modules are used in a wide variety of products. Besides portable coolers you can find Peltier modules in climate controlled car seats, scientific equipment, spacecraft and even high-end digital cameras. Peltier generators are used in the oil industry, backup power generation and to reclaim wasted heat as electricity. One really neat example is the small fire powered generator that's even being used as an emergency backup power source to charge portable devices.

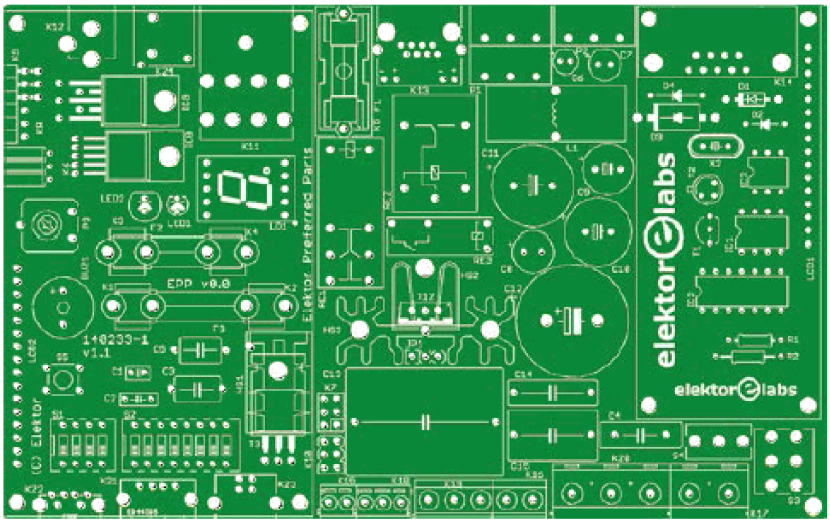
Peltier modules are easy to find commercially, especially in the surplus market, if you want to try experimenting with them. Why not track some down and give them a try?

(140045)

ELPP: Elektor Labs Preferred Parts

By **Clemens Valens**
(Elektor.Labs)

If you are old enough to remember Elektor's TUP and TUN 'one-size-fits-all' trannies then you can probably guess what this article will be about. If you don't, find out.



Given that TUP stands for 'Transistor, Universal, PNP' you should be able to figure out TUN easily. Back in the 1970s and early 80s it was Elektor's solution for specifying transistors in circuits where the exact type of transistor did not matter much. As long as you substituted a not-too-bad NPN-type small signal transistor for a TUN (typically, BC547), and a PNP-type for a TUP (BC557), the circuit would work as expected. Things have moved on since. Although my transistor guide from that era lists thousands of transistors, there are even more today. The same for many other electronic components. Today's circuit designer stands knee-deep in a lake of operational amplifiers, diodes, capacitors, microcontrollers and other (integrated) components. He/she is like a whale sifting through electronic plankton trying to filter out the parts to use in his/her circuit. No wonder then that circuits published over the last decade seem to use different components over and over again; components you may not have in stock.

At Elektor Labs we struggle with this problem on a daily basis. We receive circuit designs from all over the world using all kinds of parts. From mainstream to exotic, from Japanese to obsolete Soviet parts, we have to check them all to see if we can replace them by something that is easier to get in as many countries as possible where Elektor projects are built. What's odd to a German reader is dead common to an Australian. And every time we must ask ourselves: was this part used because of its special properties or because it happened to be in the designer's junk box?

And if we replace a part, what do we replace it with? Every Elektor Labs worker has his/her favorites and personal junk box too.

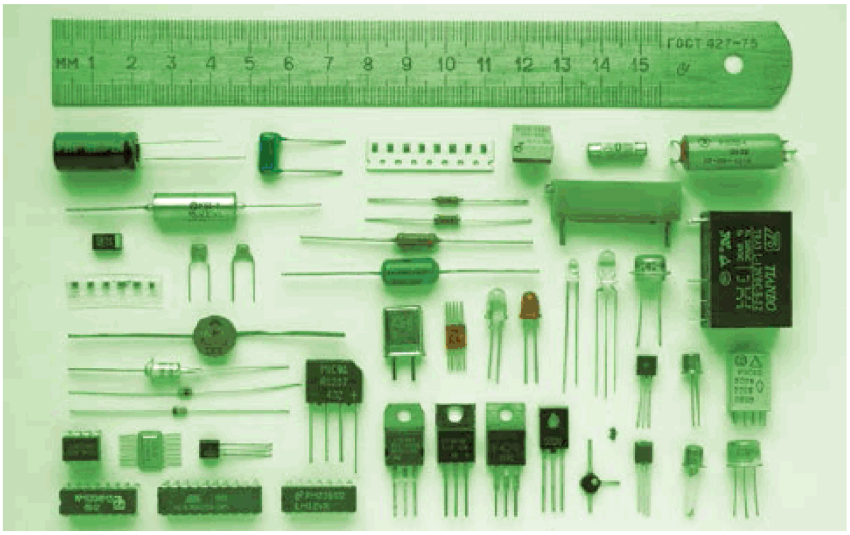
So, after many years of shoving little boxes and small bags of parts that we did not know where to put (but that might come in handy sometime, you never know—do you?), we decided to go for it and compiled a list of standard parts to be used whenever possible in Elektor projects. The advantages of such a list are manifold:

- Simplify and speed up the design process by not checking hundreds of alternative parts and by not creating new CAD library parts;
- Reduce the number of lines on the Bill of Materials (BOM), simplifying stock management;
- Allow buying in volume to reduce costs. Many distributors only let you buy parts in multiples of 10 or 50 anyway, so why not go for the whole bag instead of just the one you need?
- Always have (most of) the parts you need on stock, speeding up prototyping and building;
- Trusted footprints avoid respinning PCBs;
- Know in advance where your parts can be obtained and under what reference number;
- Know the exact specifications of each part so that a replacement can be chosen carefully instead of being guesstimated.

Of course this is not a revolutionary concept and very likely it's in place many private labs, but to implement it in such a way that it can satisfy the needs of most of our members is another thing. The hardest task is to keep the list concise. Every designer wants his/her pet parts on the list, but we only want 'universal' components. Then there is the availability requirement. We want the preferred parts to be readily available around the globe. Another daunting task is creating a CAD library with proven footprints. Arguably it took some time to compile the ELPP list and library and check it for errors, but here it is, the—Tadaa!—Elektor Labs Preferred Parts (ELPP) list.

To be honest, the list is not exhaustive because we have not included SMT parts. ELPP version 1.0 only contains through-hole (TH) parts, SMT parts will be added in v2.0.
In the list you will find components suitable for most Elektor projects: from low voltage DC to AC line connected. Most discrete parts like diodes and transistors come in three power grades: low, medium and high (where high does not exceed 100 W). Integrated circuits are not on the list except for opamps and voltage regulators. Capacitors are all at least 50 V types. Resistors cover the complete E12 range, but capacitors do not. There is only one inductor on the list and it is only there to support a switched-mode voltage regulator. Microcontrollers are not on the list, but supporting parts like quartz crystals and sockets are. The vast majority of parts has a Newark/Farnell as well as an RS Components order code.

Every part is included in our Eagle ELPP CAD library that can be downloaded from our website [1]. A DesignSpark PCB library will follow soon. All the footprints have been verified and have been designed with hand soldering and home assembly in mind. This means extra-large islands and standardized metric hole diameters. RS Components and Newark/Farnell order codes are included, greatly simplifying the creation of your shopping list. Furthermore, because every component value is in the library, you can skip the Add Component Value step during schematic capture, saving time (unless you want to draw a really pretty schematic). No more forgotten values, but also no more incomplete BOMs because sockets, fuses, jumpers and other BOM items can be found in the library too. No more funny component references, because the library follows the standards adopted by Elektor Labs for naming components.
To profit the most from the Elektor Labs Preferred Part or ELPP library you may have to change the way you're accustomed to design a little. Instead of plunking a random part on your sheet, first look through the ELPP library to see if you can't use something from it instead. Maybe that 5.6-µF capacitor you calculated can be replaced by a 4.7-µF or a 10-µF type. Often it can. Okay, that



1000-V diode may seem overrated in your 3.3-V circuit, but does it do any harm? It's a 1-amp type, readily available, and it's the same size as a 100-V type.

Think standard parts and simplify your life.

Of course the ELPP list does not limit the designer in any way. If a component is needed that is not on the list, feel free to use it anyway. Also, the ELPP list is not written in stone. This is the first version and we may have overlooked something or made a mistake. Sometimes components become obsolete or enhanced ones are discovered forcing the ELPP list to evolve. Come what may Elektor Labs will use ELPP parts whenever possible in their projects.

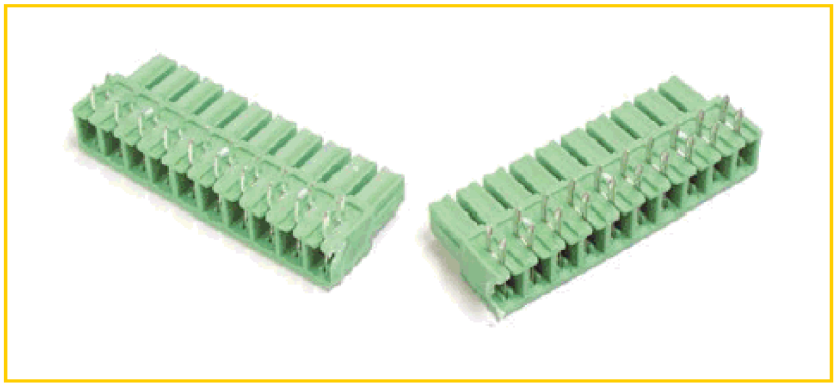
(140233)

[1] ELPP Eagle library, Excel file, datasheets: www.elektor-magazine.com/pub/Elektor%20Labs/elektor_labs_preferred_parts_elpp/

Inexpensive MyDAQ Connectivity

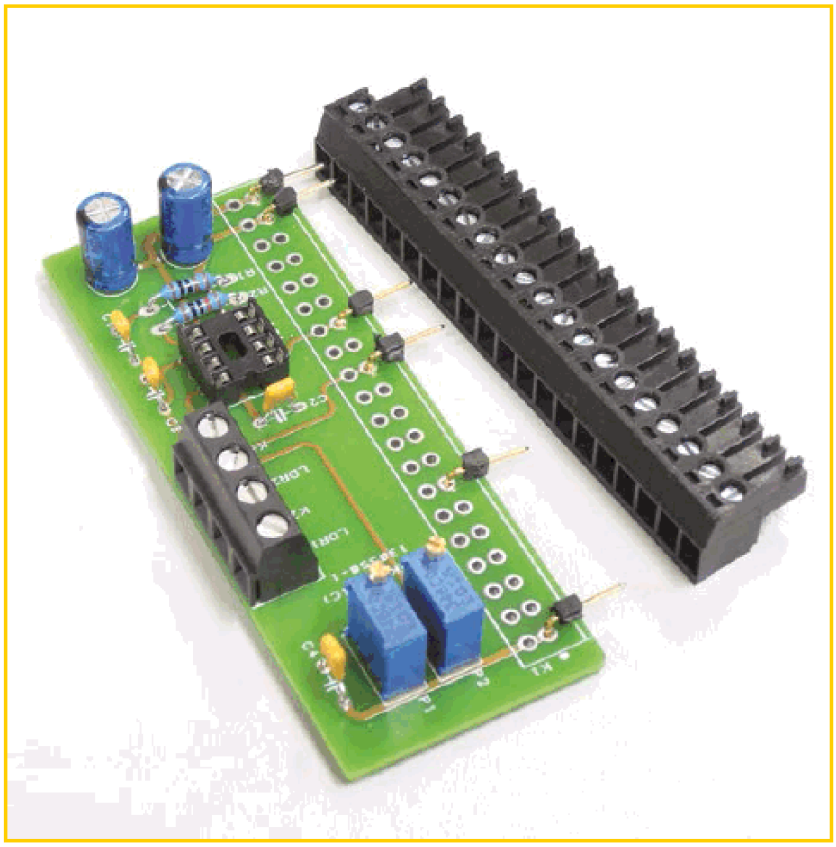
By **Thijs Beckers**
(Elektor.Labs)

Our July & August 2014 double edition featured a circuit for an Optical Theremin based on National Instruments' MyDAQ data acquisition unit and LabVIEW software. The circuit we published suggests the use of a terminal block header with a 3.81-mm lead pitch that was queried by a number of readers, mostly German. Unsurprisingly it's the .15-inch lead pitch of the internal myDAQ connector.



Due to the poor availability of this 20-pin connector and the deadline looming for the Summer Edition, we slapped up a homebrew version of the header using two 10-pin 'Pluggable terminal blocks' from Phoenix Contact (part # 1862658). The illustrations in the July & August article show this header.

These connectors tend to be quite expensive, at around € 7 (US$ 10) each. And you need two of them—not exactly an economical solution. Plus you need to file off a little from the sides of the plastic casing in the middle to make them fit close enough to each other (see small photo).



Isn't there an alternative solution? Sure. In good labs fashion the idea came to us after the deadline for the Summer Edition article had passed. But we didn't want to keep it from you, so here it is. As you can see in the photograph, we also built up a prototype using the original MyDAQ connector (the big black one in the photo) that comes enclosed with each MyDAQ. So it's free—sort of.

However, that connector is a screw terminal block hence not suitable for soldering onto a PCB. So how can it be connected to a board? Here's your answer: The clever use of a right-angle pinheader enables the MyDAQ connector to be screwed to the side of the PCB. Happily the pins are just long enough to be inserted and held by the terminal block.

Simply cut single pins from a right-angle .1-inch (2.54 mm) pitch pinheader and solder them onto the PCB where you need them. Our Optical Theremin add-on board only uses six connections to myDAQ, so that's how many we installed. You can use more if you want to ensure a stronger hold of the connector, but that's not really necessary.

(140047)

VirtualBench

All standard measurement functions in one compact unit

By **Harry Baggen**
(Elektor Dutch) and
Luc Lemmens
(Elektor Labs)

With its VirtualBench, National Instruments introduces a completely new type of measurement instrument, which combines five instruments in a compact enclosure. The operation happens completely via a computer or tablet. Elektor recently had the chance to try out one of the first available units.

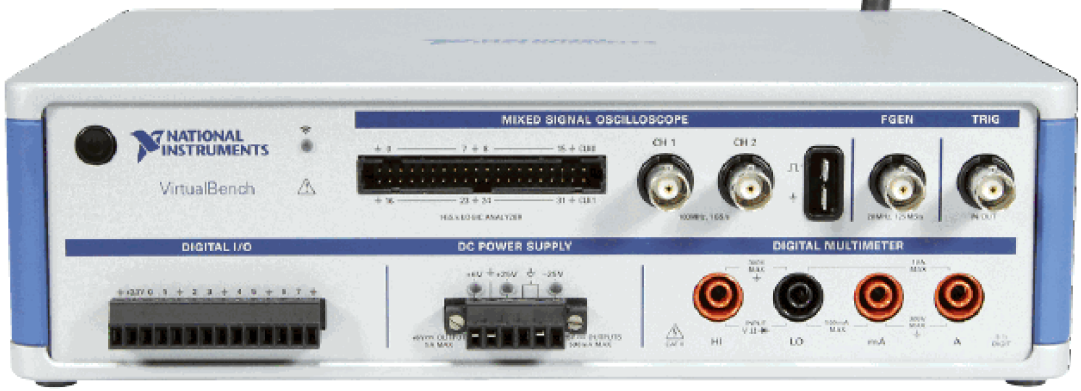


Figure 1.
The small enclosure of the VirtualBench has a multitude of connectors.

Every electronics engineer will have a number of standard instruments on the workbench, which are constantly required. These usually include a multimeter, a power supply and an oscilloscope. Manufacturers of measurement instruments have capitalized on this in the last few years by bringing multi-functional instruments onto the market, such as oscilloscopes that can also function as a logic analyzer, function generator and multimeter. In all cases these have been self-contained devices, complete with a display and operating buttons.

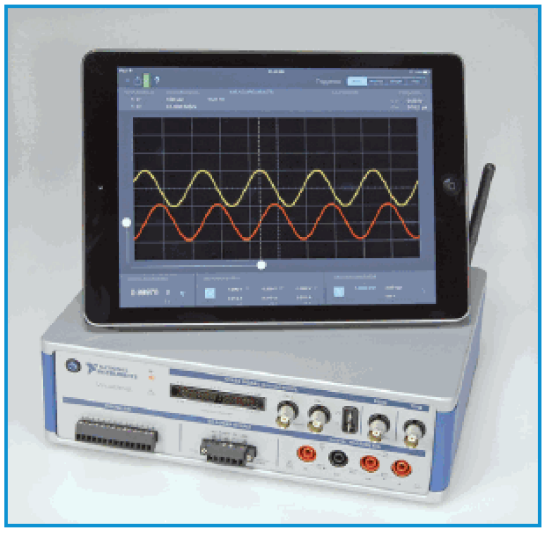
National Instruments is the big name behind LabVIEW and a wide range of professional (industrial) measurement devices and modules. It doesn't immediately come to mind as a manufacturer of 'mainstream' measurement instruments, but this is the direction it's taking with its new VirtualBench measurement instrument. NI has applied a very different philosophy to the design, compared with the products that other manufacturers of measurement instruments have available right now. VirtualBench uses the functionality of a PC or tablet for its display and operation, meaning that it doesn't need a built-in screen or controls. This idea is certainly not new, but it has never been taken as far as in this device. NI has put a dual channel oscilloscope, a 32-bit logic analyzer, a power supply with three independently adjustable outputs, a function generator and a 5½-digit multimeter into a single unit, which can all be operated via one common interface.

When National Instruments asked us if we wanted to test one of the first VirtualBench units available in Europe, they didn't need to ask twice! On paper, the VirtualBench appears to be a fantastic combination of instruments, but how well will it perform during normal use in our labs? This was something we wanted to find out for ourselves, of course.

Enclosure: small and strong

Although the dimensions of the VirtualBench are specified on National Instruments' website, it still came as a surprise how small it was when we saw it for the first time. It seemed not much bigger than a lunchbox. When you consider how many standard devices the VirtualBench replaces, you'll realize that it saves a lot of space on the workbench.

The instrument comes complete with all the necessary cables: two scope probes, two multimeter probes, a 40-way connector with probes for the logic analyzer, a power cable, a USB cable (with thumbscrew fixing) and a screwdriver.
The enclosure is quite strong, with aluminum top, bottom and front panels; the rest of the case is made out of sturdy plastic. At the front are all the signal connectors, as well as the on/off switch (**Figure 1**). The connections for the digital I/O and the power supply are made with screw terminals, which makes it easy to connect wires. At the back you'll find the power input, a USB socket and a WiFi antenna. The device has a small fan for cooling, which fortunately is hardly audible.
We've already mentioned that the VirtualBench should be used in combination with a (Windows) computer or a tablet (at the time of writing, only the iPad is supported). The connection to the PC or laptop is made via a USB cable, the built-in WiFi host or via an existing WiFi network. The communication with tables is wirelessly wihout exception.



The VirtualBench can also be used in combination with an iPad.

The software:
everything in one window

In practice, there is no need to install any software on the PC or laptop. When the VirtualBench is switched on and connected to the computer via the USB cable, the computer detects the USB device and automatically runs the program that it finds in the flash memory of the VirtualBench. Several USB and HID drivers are installed when it is connected for the first time. You may also have to give permission to the Windows firewall to let the program access the computer. Subsequent connections will be a lot quicker. The computer needs about 15 to 20 seconds to load the program. Things are quicker on an iPad; once the program has been installed via the iTunes Store, the iPad only needs to log in on the network of the VirtualBench.
In **Figure 2** you can see a screendump of the Windows version of the VirtualBench program. All of the instruments are shown here in one

Specifications

2-channel oscilloscope
- Input bandwidth: 100 MHz
- Sample rate: 1 Gsample/s (500 Msample/s for 2 channels)
- Buffer size: 1 Msample/channel

Logic analyzer
- Number of channels: 34
- Max. input frequency: 100 MHz
- Input voltage: 0 to 5 V

Function generator
- Max frequency: 20 MHz (sine wave), 5 MHz (square wave)
- Waveforms: sine, square, triangle, DC

Digital multimeter
- Display: 5½ digits
- Basic accuracy: 0.015% (V_{DC})
- Measurement functions: V_{DC}, V_{AC}, I_{DC}, I_{AC}, resistance, diode, continuity
- Max. voltage/current: 300 V/10 A

Adjustable power supply
- Number of outputs: 3
- Channel 1: 0 to 6 V/1 A max.
- Channel 2: 0 to +25 V/0.5 A max.
- Channel 3: 0 to -25 V/0.5 A max.

Digital I/O
- Number of channels: 8
- Configurable as inputs or outputs
- Outputs 3.3 V, inputs 3.3 V/5 V compatible

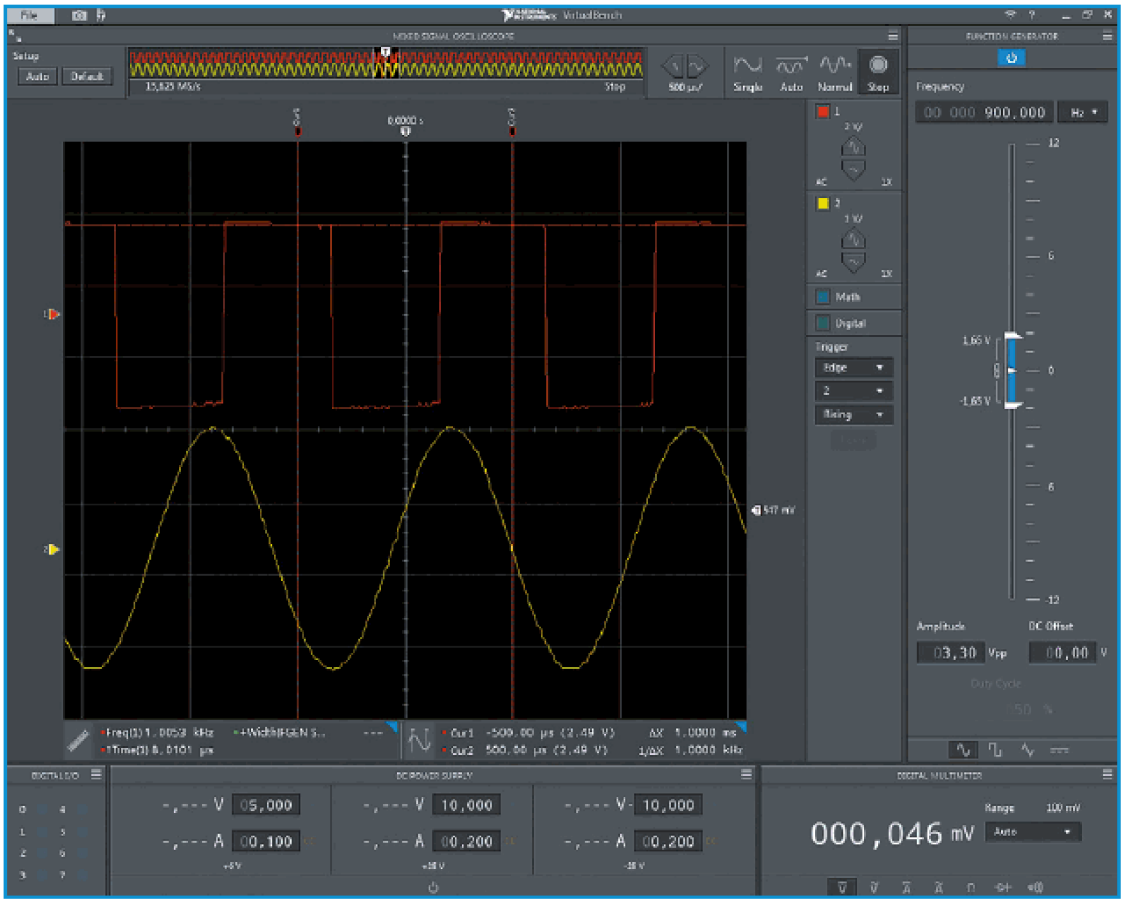


Figure 2.
Screendump of the VirtualBench program (Windows version).

window. The scope/analyzer section along with its controls and display take up the most space, making the measured signals clearly visible. The other instruments have been arranged around the scope. To the right is the function generator with a slide control for setting the DC offset and the amplitude of the signal. Below this is the multimeter. To the left are three boxes for the power units, and the digital I/O section is at the far left. The operation is very simple and hardly needs any explanation. The operation is almost entirely via the mouse. Values for the frequency, amplitude or voltage can be input in several ways, such as directly via the keyboard or by scrolling the mouse. When a value is outside the permitted range of a particular instrument you'll get a warning and the value will be ignored. If you click on the top-right corner of most sections, you'll get a sub-menu with some other settings and extra information. The main toolbar has menus for settings and help, as well as a useful screenshot function. There is also a button that saves the measured data as a CSV file.

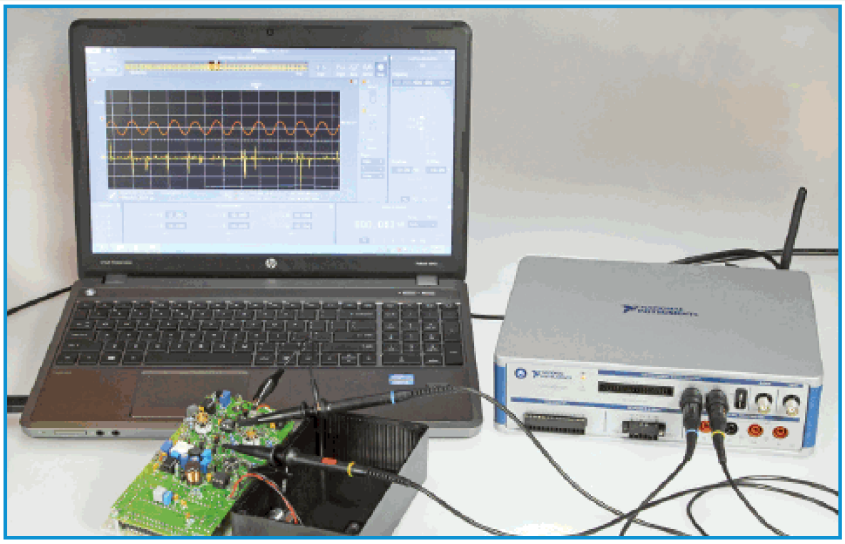
The iPad version has almost the same functionality, but is obviously operated via the touchscreen. In some instances this is easier than using a mouse, but in others it won't be.

In the lab

When we used the VirtualBench in the Elektor lab we tried controlling it via an iPad, but we found that the app didn't come across as very 'grown up'. Our preference was therefore to operate it via the PC. This is obviously a personal preference, but we found it very useful to be able to see the control window with all the settings and measurements on a large monitor while we were working on a circuit. The following comments therefore apply to the Windows version of the software.
It won't take long to get used to the VirtualBench. The connections on the device are self-explanatory and are identical to those that you would find on the individual measurement instruments. Some of the connectors are somewhat close together. We thought that there should have been more room between the individual BNC connec-

tors, as well as between the BNC connector and that for the logic analyzer. Then again, it may be that it takes some time to get used to it. At first the screw terminals used on the power supply section seem like a good idea, but after a while you start to miss those practical banana plugs that are found on most lab power supplies (perhaps we could design an adapter board for this). You can obtain spare connectors for the power supply and digital I/O from NI, which could be useful if you use them constantly.

Initially, you may have to look around to find all of the functions and settings in the software, but most of them have been laid out logically and are where you would expect them to be. We hardly had to look at the manual (we're experts, after all!).



Oscilloscope/analyzer

The scope section takes up the most space on the screen. This is to be expected, since you want to see the measured waveforms clearly. If that isn't enough, you can make the scope window fill all of the space on the screen. An auto-setup button ensures that you can quickly get a good display of the measured signal. The scope section offers just about all of the settings and trigger functions that are found on an ordinary scope. Along the top is a representation of the measurement buffer. Using the mouse, you can select any section of this buffer and display it. There are two bars below the scope screen from where you can control cursors on the screen and carry out various measurements (there are over 20 different ones) on the signal. You can also carry out mathematical operations on the signal using the Math button (at the moment this is limited to addition, subtraction, multiplication and a standard FFT function with relatively few settings). The Digital button is used to select the digital channels and the corresponding trigger action. When we were using the scope it appeared to react very slowly to changes in the signal. We found out that this was due to the default setting of '32 Averaged' for the signal acquisition. This seemed to be too much of a good thing, and we set it down to '2 Averaged' and eventually to 'Sample', which finally gave the feeling that you were watching a real-time signal. You should save the new configuration, otherwise the instrument will always start with the same default settings. One thing we missed on the scope display was the value for the vertical and horizontal scale. At the moment you have to look at the (fairly small) values written next to the controls. And there is enough room left on the screen…

When it's used as an analyzer it's easy to see several signals, but with 32 channels the screen gets too crowded and you wish you had a real analyzer with advances analysis functions. But this is a problem you get with all combined scope/analyzer devices. Incidentally, it's quite tricky to make sure that all of the measurement cables (delivered without terminals, unfortunately) from this 40-way connector are connected properly to the circuit under test.

Function generator

The function generator (14-bit, 125 MS/s) works as expected and outputs all the standard waveforms with a frequency range that's more than enough for most applications. An unusual feature is to have a DC voltage at the output (max. ±6 V into 50 Ω, or 12 V into 10 kΩ), which is not found in many instruments. The output voltage can be set over a wide range. The slide control was found to be very useful for this, as you can use the mouse to quickly set the offset and signal amplitude. In principle, you should be able to program your own waveforms (AWG) with the signal generator, but this function is not (yet?) available in the software. Users can write a LabVIEW VI though for generating custom waveforms.

Multimeter

The multimeter is also very good, with a decent accuracy. It has all the features found on a standard multimeter. However, we miss the facility to increase the size of the multimeter section on the screen. After all, when you're taking measurements with the multimeter it is often useful to display it on the screen in a larger format. This comment really applies to all of the instruments on the screen: Everything has a fixed position and this cannot be changed. The scope is obviously the most important and uses most of the space on the display. The other sections of the screen can't be moved, nor can they be increased in size. This is a pity, and hopefully this is something that will be rectified in a future release of the software.

Power supply

The lab power supply uses a limited area in the window, but still compares well with a standalone supply. Perhaps not as far as the output current is concerned, although the output power is sufficient for most small circuits. The outputs can be set very accurately. For each of the three channels you can set the output voltage and the maximum output current. On the screen you can see the measured values of the output voltages and currents. This is a great feature, which is sadly lacking in many other lab power supplies! The output connector wasn't that brilliant in daily use, as we mentioned in the description of the hardware.

Digital I/O

We couldn't immediately think of an application for this function, although we're sure they exist. When you use VirtualBench in combination with your own LabVIEW program it should prove very useful.

Conclusions

There are still many functions and features that we haven't covered, but this article is certainly enough to give a good first impression of the instrument. When reading this article you may feel that we were finding a lot of faults with the VirtualBench, but the opposite is true: We were blown over by it and many of our editors and engineers at Elektor would love to have such a device for their workbench, and ideally one for use at home as well.

It is an ideal combination of instruments with specifications that are more than sufficient for most applications. Most of our remarks were about the operation of the software. It should be noted that these are things that could easily be rectified in future releases of the software. We should also remind ourselves that this is a new product, with new software. We are therefore very curious to see what new features will be found in the software in a year's time! NI is a company that has always listened to the remarks and wishes of its customers. We have certainly been convinced by the product and the concept.

It is perfect for standard lab use, but also for schools and even hobbyists, although the price could be an obstacle for the latter group. But we think that VirtualBench is worth every cent of its $1999 / €1690 cost!

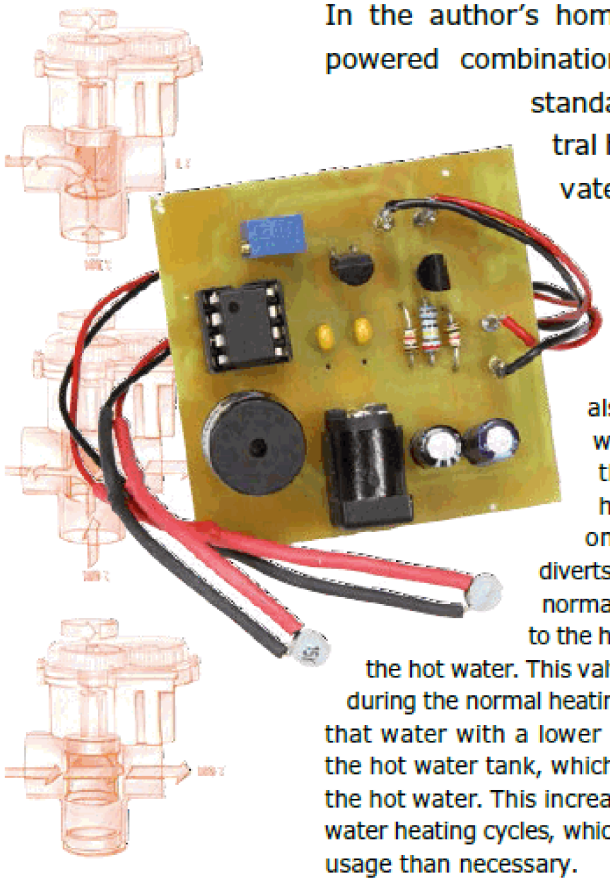
If NI wants their instrument back, they will have to come and get it.

(140252)

Web Link

www.ni.com/virtualbench/

Three-way CH Boiler Valve Monitor



In the author's homeland natural gas powered combination boilers are the standard source of central heating (CH) in private homes.

These highly efficient apparatus not only provide heating via the radiators, but also take care of the hot water for the kitchen and the bathroom. When a hot water tap is turned on, a three-way valve diverts the hot water from the normal central heating circuit to the heat exchanger/tank for the hot water. This valve can sometimes leak during the normal heating cycle, with the result that water with a lower temperature is fed to the hot water tank, which therefore cools down the hot water. This increases the number of hot water heating cycles, which leads to a higher gas usage than necessary.

When the hot water tank is being warmed up, the temperature of the feed pipe to the tank should be higher than that of the return pipe. During normal operation, the temperature of both pipes should drop at the same time. With a leaking valve, the feed pipe can be at a substantially lower temperature than the return pipe. This can be clearly seen from the graphs that show the

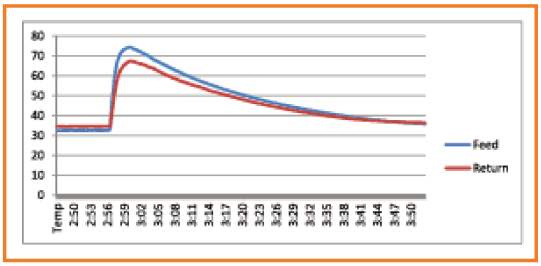


Figure 1. When the three-way valve functions properly, the temperature of the Feed will always be above that of the Return.

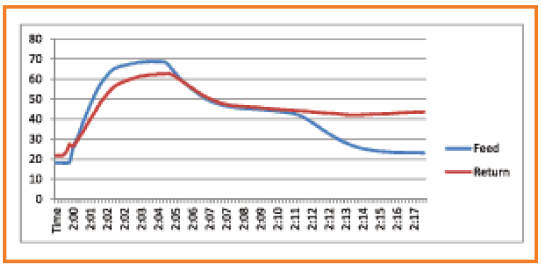


Figure 2. When there is a defect the temperature of the Feed can drop below that of the Return.

Based on an idea by
Sybe Sijbesma
(The Netherlands)

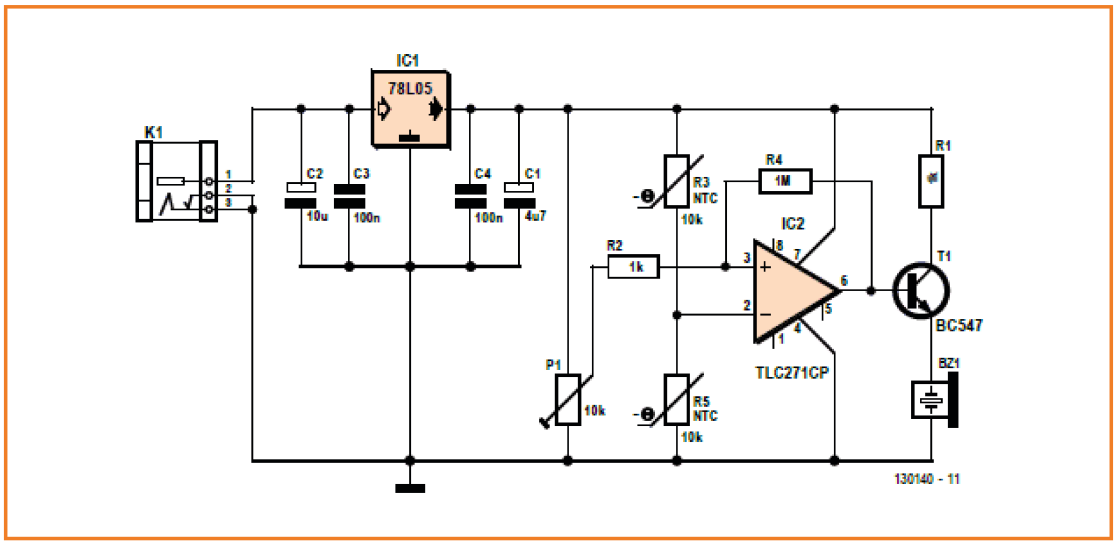


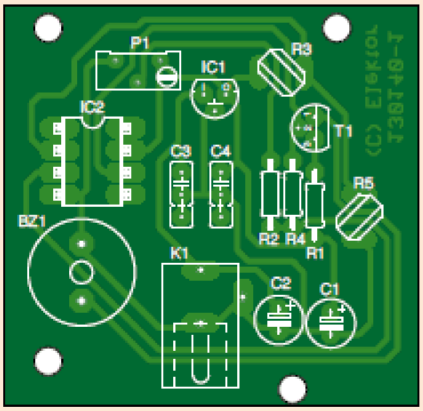
Figure 3.
This circuit compares the two measured temperatures and sounds a buzzer when the temperature of R5 drops too much below that of R3.

Component List

Resistors
R1,R2 = 1kΩ
R3,R5 = NTC 10kΩ
R4 = 1MΩ
P1 = 10kΩ trimpot, vertical

Capacitors
C1 = 4.7µF 25V radial
C2 = 10µF 25V radial
C3,C4 = 100nF

Semiconductors
T1 = BC547B
IC1 = 78L05ACZ
IC2 = TLC271CP



Miscellaneous
BZ1 = actieve buzzer (i.e. internal oscillator), e.g. Kingstate KPEG-200A
K1 = adaptpr socket, PCB mount, e.g. DCJ0202
PCB # 130140-1, artwork download at [1]

measured temperatures for a good three-way valve (**Figure 1**) and a leaking valve (**Figure 2**).

When NTCs are mounted onto the feed and return pipes for the hot water tank, it is possible to detect faults in the three-way valve at an early stage. The circuit shown here (**Figure 3**) stands out by its simplicity, and consists of little more than an opamp (TLC271), which is configured as a comparator to compare the temperature of the two NTCs. The threshold value is set with the help of preset P1. When the temperature of NTC R3 drops too far below the temperature of NTC R5, the comparator changes state and the buzzer is turned on via the transistor, to indicate that something is wrong. A 5-V regulator was added to the circuit so that it can be powered by any mains adapter with any output between 8 V and 15 V DC. The current consumption is less than 10 mA, even with the buzzer on.

A small PCB has been designed for the circuit (Figure 4), but it can just as easily be built on a piece of experimenter's board. The NTCs should be mounted close to the hot water tank on the feed pipe (R3) and return pipe (R5), using cable ties.

(130140)

Web Link

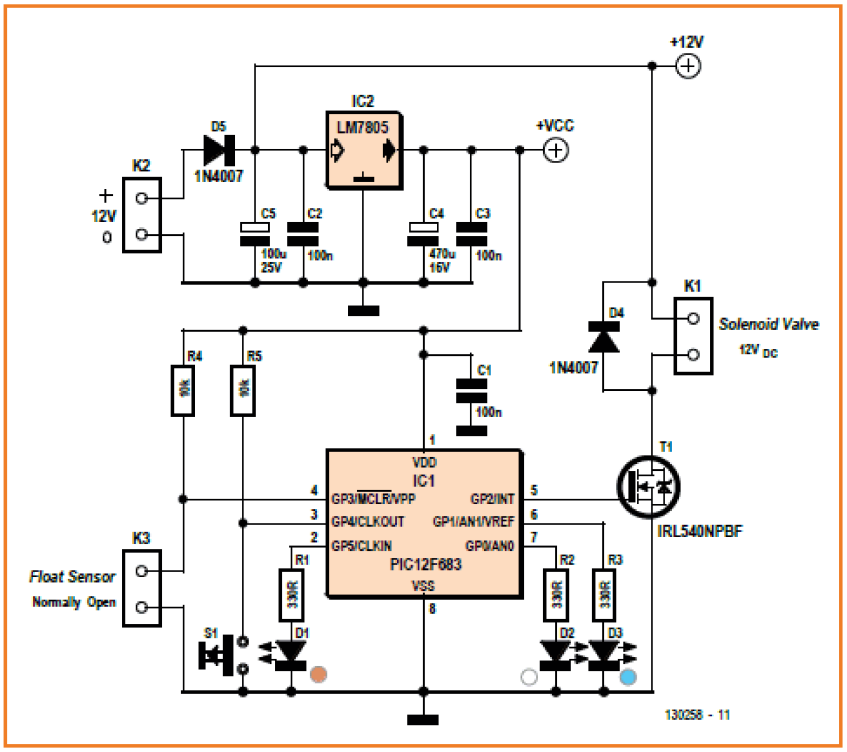
www.elektor-magazine.com/130140

Efficient Water Solenoid Valve

for a reverse-osmosis water filter

By **Maarten Vandekeybus** (Belgium)

The manual operation of the tap for a water filter is something that seems a suitable candidate for automation. This can avoid the unnecessary waste of a large amount of water. The circuit has been configured such that it uses very little power.



The author has a reverse osmosis water filter in his home, for the purification of tap water. A small tank of about 25 liters (6.6 US gallons) is connected to the filter. You always have to manually open a tap to top up the tank. However, since it can take up to four hours to fill the tank, the tap is often forgotten about and left open much too long, with the result that a lot of water is wasted. To stop this happening, the author decided to design a small circuit that would automate this process. The circuit opens a solenoid valve until a float sensor detects that the tank is full.

The author ordered a 12 V_{DC} solenoid valve from eBay ($16). It functioned perfectly well, but it used a lot of power (nearly 18 W), which isn't very

Figure 1.
The electronics in the economical water solenoid valve circuit consist of little more than a PIC12F683.

energy conscious. He remembered having seen an article in Elektor on reducing relay power consumption, based on the fact that a relay requires only a fraction of the nominal power once it has been actuated.
After some experimentation the result was a small microcontroller circuit that used a power MOSFET to drive the solenoid. This ensures that the full supply voltage is applied to the solenoid initially, after which it is lowered to a value that still keeps the valve open, but reduces the power consumption considerably. There is also a switch delay of about one minute built in to avoid the valve from opening and closing repeatedly when there is some turbulence in the water level. The circuit has been provided with three colored LEDs that show what the current operating mode of the circuit is.

Schematic diagram

In **Figure 1** you can see the circuit diagram of the author after it has been checked over by Elektor labs. A PIC12F683 has been used for the microcontroller. Output GP2 drives the power MOSFET (T1), which operates the solenoid. Once the solenoid has been actuated, the reduction of the solenoid voltage is achieved using pulse width modulation. It uses a frequency of about 488 Hz, and the mark-space ratio is 30:256. The experiments of the author showed that the total power consumption of the solenoid could be reduced to just 0.4 watts once the valve was actuated. This is a huge difference compared to the nominal 18 watts consumed when it is switched on!

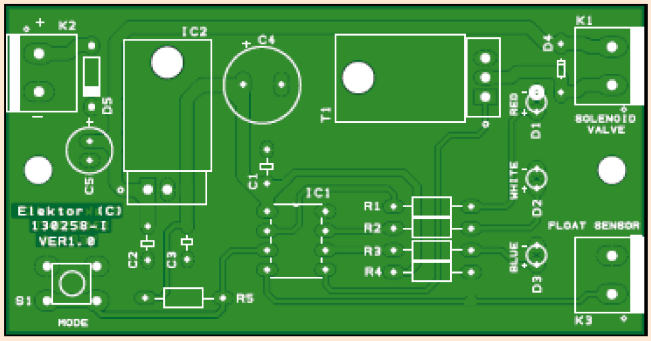
Three LEDs are used to show the current state of the circuit. The white LED (D2) indicates that the circuit is in automatic mode. The blue LED (D3) shows that the valve is open and the red LED (D1) is on when the circuit is in the 'delay state'; in this case the state of the float sensor has changed, but the valve has not yet reacted to it. The LEDs are also driven via PWM in order to save power.

Component list

Resistors
R1,R2,R3 = 330Ω
R4,R5 = 10kΩ

Capacitors
C1,C2,C3 = 100nF
C4 = 470µF 16V radial
C5 = 100µF 25V radial

Semiconductors
D1 = LED, red, 3 mm
D2 = LED, white, 3 mm
D3 = LED, blue, 3mm
D4,D5 = 1N4007
T1 = IRL540
IC1 = PIC12F683, programmed, Elektor Store #130258-41
IC2 = 7805



Miscellaneous
K1,K2,K3 = 2-way PCB screw terminal block, 5mm pitch
S1= pushbutton
PCB # 130258-1, see [1]

Figure 2.
There is plenty of space on the board, which makes it suitable for less experienced constructors.

Switch S1 is used to switch the circuit to one of three operational modes: Manual, automatic and off:

- Off mode: In this state the circuit is inactive and none of the LEDs is on.
- Manual mode: The valve is turned on continuously. In this state both the white and blue LEDs will be on.
- Automatic mode: The valve is turned on and off depending on the state of the float sensor. The float sensor used here is normally open and closes once the water rises above the float level. Ripples on the water could cause the float sensor to change state rapidly, and hence turn the valve on and off repeatedly. To avoid this, the valve only changes state after a one-minute delay following a change in the state of the float sensor. The red LED will be on while this delay is taking place.

Figure 3.
The solenoid valve and float sensor used by the author.

When the power supply is turned off (either deliberately or due to a power failure) the current state of the circuit will be stored in the internal EEPROM. Once the power is restored the circuit will resume in the same state.

It's best to use a power adapter with a 12 V DC output (at about 1.5 A) for the supply. Diode D5 is for reverse polarity protection. IC2 turns the 12 V into a stabilized 5 V DC for the PIC.

A simple board

The printed circuit board shown in **Figure 2** has plenty of space and contains only standard components. The construction should not give you any problems. Make sure that you've programmed the PIC before you solder it onto the board. You can freely download the source code and hex files from the Elektor Magazine website [1]. Alternatively, order a ready-programmed PIC from the Elektor Store (130258-41).

It's recommended that the whole circuit is mounted in a splash-proof enclosure, such as the one shown at the start of the article. Such enclosures should be available from most DIY stores.

(130258)

Web Link

[1] www.elektor-magazine.com/130258

Taking the Strain

Integrating precision strain gage measurements into a programmable SoC

By **Kendal Castor-Perry** and **Nidhin MS**
(Cypress Semiconductor)

Many pressure and force sensors rely on resistive sensing elements connected in some variant of the so-called bridge configuration. In the electrical context, a 'bridge' is a common topology in which four two-terminal devices are connected in a loop. Think of the ubiquitous 'bridge rectifier' — if you are reading this article, chances are that you have used one of those in a power supply design.

When the two-terminal devices are impedances whose values can be changed by some external influence, you get an incredibly versatile sub-circuit. If you mix impedances with difference frequency behaviors, you can create interesting filter and oscillator circuits— you've probably heard of the Wien Bridge oscillator [1], which relies on the frequency-dependent transfer properties of **Figure 1**.
Moreover, bridges turn up in other passive filter networks too, except that they are often drawn differently, with the connections 'twisted' around, and then called lattices instead. In **Figure 2** (also from Wikipedia), treating the left hand nodes as inputs and the right hand nodes as outputs, and make the Z branches capacitors and the Z' branches resistors, you get a first-order all-pass filter.

Real-world issues

The bridge devices we want to focus on here are typically made of resistors whose values change under the influence of some physical parameter. This could be temperature (sensed intentionally, or happening as a side effect), a magnetic field, incident light, humidity or—a huge industrial sensor application—physical strain in a mechanical system.
You may remember from mechanics that strain is what happens to a physical object when you apply stress to it—a dimensional change of the object. It might be change in length, cross-sectional area or both. This affects the structural properties of the material that the object is made of, and this can affect the passage of an electrical current through that material—it changes the resistance, in other words.

A bridge of resistances of which one or more are attached to an object in a way that their resistance is affected by mechanical changes to the object is usually called a strain gage—or gauge, in some parts of the world with plentiful supplies of the letter *u*. Often the resistors are in the form of thin metal traces on a polymer substrate that is bonded to the structure of interest. If you're an instrumentation engineer—or just a hard-pressed electronic engineer who's been asked to interface to one of the things—you'll be interested in how to extract interesting information from the voltage changes that result when such a strain gage actually gages some strain.
If this was trivial, there would be no need for how-to articles like this, and no need for semiconductor companies to make products that help you out. Nevertheless, as is so often the case in the world of analog measurements, there are pitfalls

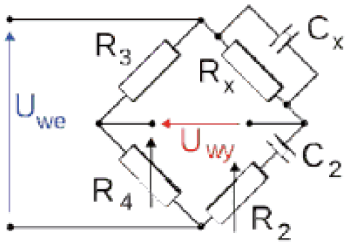


Figure 1. The Wien Bridge (reproduced from [1]).

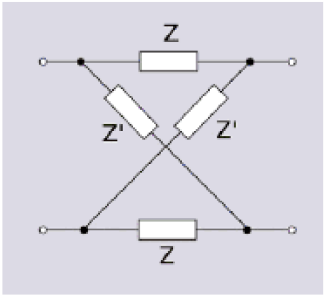


Figure 2. A bridge redrawn as a lattice.

for the unwary. That's because the strain in such a gage has to be kept quite small, otherwise the material will deform permanently. The small strain implies a small relative resistance change, and so the bridge is only slightly imbalanced as a result.

News views from the bridge

So, how do we interface to a bridge of resistors, and what might we find? Measuring the small voltage across the sense terminals of a driven bridge is a classic application for an instrumentation amplifier. The job of such an amplifier is to deliver an amplified representation of a small voltage that appears between two nodes that might both be moving around on top of a larger, unknown common mode voltage. The output of that instrumentation amplifier will be converted to digital with an ADC—practically no industrial measurement system is analog all the way through these days.

Old-school instrumentation amplifiers often employ sophisticated techniques to deliver very low levels of input offset and drift, and can command a premium price. In systems where a digital result is required, high performance delta-sigma ADCs with 20 to 24 bits of resolution can be directly deployed, without additional Preamplification, to achieve microvolt levels of precision. As sensors of all forms become ubiquitous, a trifecta of design pressures— cost, size and power—are driving designers to look at solutions that are more economical. However, the lower performance analog components usually available in modern mixed-signal microcontrollers fall short in terms of basic signal path quality. There is one technique, though, that can take you most, and usually all, of the way towards successfully implementing complete single-chip sensor systems for low-output bridge-based sensors, and that is *Correlated Double Sampling*.

Now, the analog performance of the latest programmable system-on-chip devices is certainly a step up from common analog-equipped microcontrollers. Factory-trimmed amplifier offset voltages of better than a millivolt are now available in economical devices such as Cypress Semiconductor's latest PSoC 4200 family. For many applications, such devices can be considered for simple, familiar configurations, and can significantly shorten the design cycle for many smart sensor and process monitoring applications.

However, when signal level changes fall below millivolts towards microvolts, even devices at the state of the commercial art need a little help from smart system design. Fortunately, such programmable SoC devices include flexible analog routing structures, and this enables the use of a great precision-enhancing technique: correlated double sampling (CDS).

CDS is an umbrella term for a range of techniques in which several measurements are combined in a way that mitigates some sort of error that is correlated (hence the term) across those measurements.

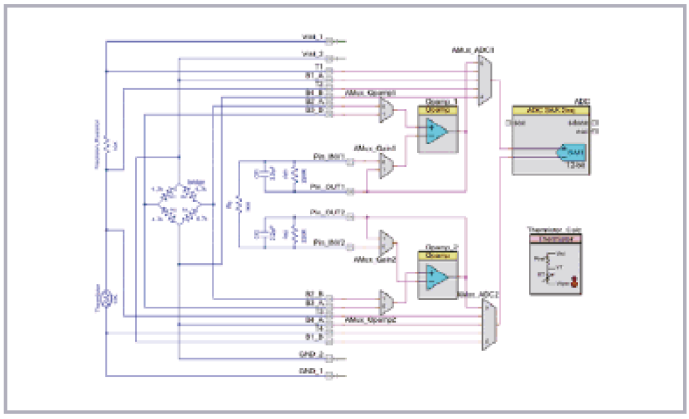


Figure 3. A circuit using CDS for both differential bridge and single-ended ratio measurements.

This simple technique—measuring a floating voltage twice, using a differential input connection that is flipped in polarity between measurements, and then subtracting the two conversion results—can eliminate the majority of errors that an analog front end can introduce. Inherent static input offset voltage is cancelled out, as is any very low frequency input noise that appears, whatever its cause (supply influences, thermal effects and LF noise from amplifier input devices) over the time period of the two measurements, as a static input voltage error. Finite common mode rejection effects (whether linear or non-linear with respect to the common mode voltage) are also eliminated at least to first order, because the primary impact of sensitivity to the common mode component at the two monitored nodes is to slightly change the apparent offset voltage of the input channel.

For more advanced applications where some of the error signals vary more rapidly in the time domain, an extension of the technique can be used to eliminate the consequences of sampling the normal and reverse input connections at different points in time. You can read about that 'Filter Wizard' technique at, for instance, [2]. The electrical circuitry is the same; it's just the digital post-processing that gets more sophisticated.

Onto a PSoC

As a practical example of this technique, consider the schematic in **Figure 3**. This was implemented on the recently introduced PSoC 4200 mixed-signal system-on-chip from Cypress. Such devices are differentiated from conventional mixed-signal microcontrollers through their more comprehensive signal routing and switching capabilities, and these have been taken advantage of in this example. Let's focus first on the block of circuitry that processes the output from the bridge transducer. Analog multiplexers select several different sets of signals for conversion. A pair of opamps are configured as a differential amplifier whose gain can be switched between unity (for diagnostic purposes) and a high value set by exter-

nal precision resistors. The performance of these amplifiers is good, but not sufficient to achieve the microvolt-level result stability that we'd like for such a transducer. Therefore, the CDS process is carried out using signal switching right at the amplifier inputs, meaning that residual offset, drift and low frequency noise is cancelled out from the final calculated measurement, as already described.

All set to calculate

To calculate the strain in the bridge, we need to have two pieces of information. First, we need the high-quality measurement of that small voltage between the two output nodes of the bridge. In addition, we need to know what voltage is being applied to the excitation nodes of the bridge. Here, that voltage is applied by GPIO pins on the SoC (so that the transducers can be disabled to lower power consumption). A Kelvin (four-wire; force & sense) connection routes the drive node voltage into the analog multiplexing so that the ADC can capture these voltages too. They can be very close to the supply rails, so the (rail-to-rail capable) ADC is configured to use the power supply rail as its reference. This does not affect the accuracy of the strain calculation because the reference value is a constant that cancels straight out.

Note that to calculate the overall strain, we do not need to know how many resistors in the bridge change their value when stress is applied to the structure that the sensor is fixed to. If we want to calculate the actual stress in the sample that caused it, though, we need to know more about the construction of the bridge. That's because for the highest accuracy, a small linearity correction is applied when the calculation is made, and this correction method depends on how many resistors are strained and how many are static reference elements.

High performance is available from this configuration. The internal ADC of the PSoC used in this example has an intrinsic resolution of 12 bits, but here we can take advantage of the high maximum sample rate (up to 1 Msps is possible) and use the built-in hardware averaging (no CPU work required) to push the noise floor down and thereby increase the equivalent resolution. With the 44x input differential amplifier and 256 averages, measured results indicate an equivalent performance of close to 15 bits RMS right at the bridge output terminals, equivalent to an RMS noise level of under 8 microvolts, with a DC error that's very difficult to measure using

Software to the rescue

It is tempting to think about using the differential-input measurement channel to acquire the voltages of interest. However, in fact this is unnecessary. Instead, it suffices to measure the voltage on each end of each of the resistors, using a single-ended measurement channel. Subtracting the two readings gives you the voltage across the resistor. Of course, we know that subtraction of two nearly equal quantities can produce some uncertainties in a result, and we do need to make sure that our converter resolution doesn't result in a loss of numerical precision. Measured results on the front end in this mode indicate an RMS equivalent number of bits ('ENOB') well in excess of 16 bits.

The big advantage of this method, which is another form of CDS, is that any static or quasi-static error at the input of the measurement channel gets cancelled out, even though only single-ended measurements are taken. So the result is essentially unaffected by channel offset or low frequency noise, as in the bridge measurement case. It's a great technique for getting good performance out of a single-ended measurement channel. It is still dependent on the reference voltage used by the ADC—but that term gets divided out too, when we ratio the voltages across thermistor and precision resistor.

The result is a very system-insensitive measurement of a resistance ratio. This can be converted to a temperature value in software. The figure shows another elegant development afforded by the tools that support the SoC used here—the temperature conversion is done by a software component, included in the development tool, which is simply dragged onto the schematic and configured for the thermistor in use.

In this example, the ADC measures eight different voltages from the bridge and thermistor circuits. Between the measurements, firmware controls the analog multiplexers to select the next voltage to be measured. The ADC in PSoC 4200 device has a digital post-processing block that can be used to average multiple ADC counts to yield a higher effective resolution. This averaging process is independent of the CPU. In power-sensitive applications, the CPU can spend most of the time in Sleep mode. The ADC wakes up the CPU after each measurement (including the averaging process). The CPU can then control the multiplexers to select the next voltage, initiate the ADC conversion, and go back to Sleep mode. After completing the measurement of all eight voltages, firmware can calculate the bridge strain and temperature.

The programmable system-on-chip used has a variety of output options to show the calculated results. It can provide analog outputs using IDACs (which are current sourcing/sinking digital to analog converters), directly drive segment LCDs, or communicate the results through I^2C, UART, SPI or any custom communication protocol implemented using the programmable digital logic.

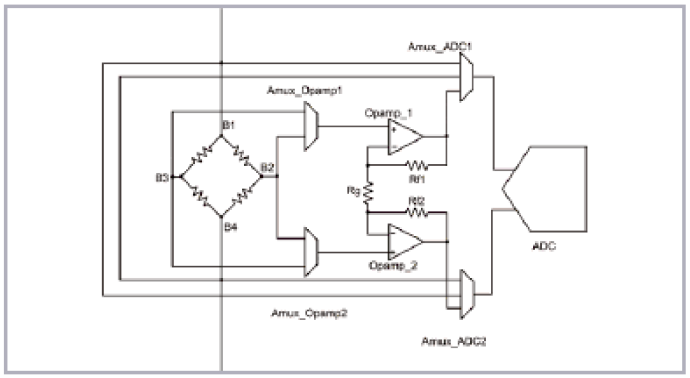


Figure 4. Detail of the bridge measurement part of Figure 3.

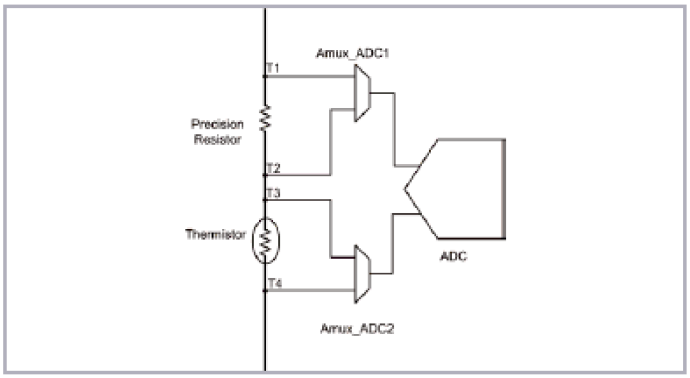


Figure 5: Detail of the resistor ratio measurement part of Figure 3.

ordinary equipment. This is achieved with no special considerations for board layout.
Figure 4 shows the simplified diagram of the bridge sub circuit. The relative bridge strain is the ratio of bridge output to bridge excitation. Therefore, calculating the relative bridge strain requires measurement of voltages B1-B4 and B2-B3. For CDS, these voltages need to be measured twice, flipping the polarity. The ADC measures the excitation voltages B1-B4 and B4-B1 directly. Bridge outputs B2-B3 and B3-B2 are measured after the amplification provided by the differential amplifier.

The firmware then calculates the relative bridge strain as:

$$strain = \frac{ADC\ count(B2,B3) - ADC\ count(B3,B2)}{Gain_{preamp}(ADC\ count(B1,B4) - ADC\ count(B4,B1))}$$

where $Gain_{preamp}$ is the gain of the differential preamplifier. Implemented on the development board of the SoC used, this configuration provides microvolt-level stability and can be used to accurately calculate very low levels of strain. The front-end gain of 44x set in this example suits full scale strain levels of around 2%. At the bandwidths required for these measurements, significantly higher gains could be set through appropriate external resistor choice, for systems where there is low static strain in the sensor.

This example has also been equipped with a temperature measurement subsystem, operating simultaneously with the strain measurement subsystem, which also uses a form of correlated double sampling. Knowing the temperature of the transducer element itself is sometimes helpful for optimizing the calibration of the sensor sensitivity. These days, thermistors offer probably the best accuracy to cost profile of all temperature measurement solutions. Calculating the temperature requires measurement of the thermistor's resistance. To optimize accuracy, the resistance measurement must be only minimally dependent on the properties of the measurement system, such as offset, noise, and the accuracy of the reference and the conversion gain.
The basic resistance measurement idea is simple and illustrated in **Figure 5**: pass a current through the thermistor and a precision resistor, measure the voltages across both the thermistor and that resistor, and the ratio of those voltages is identical to the ratio of the resistances. The accuracy is obviously directly related to the accuracy of the precision resistor, but these days that is a very affordable component. However, it is also dependent on the fidelity with which we can capture the voltages across the two resistances, more about this in the **inset**.

Conclusion

Correlated double sampling can bring impressive levels of precision to the measurement of small transducer output voltages that you might not have thought were within reach of cost-effective microcontroller products. It's especially effective when implemented using the versatile analog routing capability afforded by the programmable system-on-chip devices used in the example shown. The easy-to-use measurement subsystems described here make it possible to create small, low cost yet high performance smart sensor front-ends that can be customized to your own needs and brought to market rapidly.

(140230)

Web Links

[1] Wien Bridge: http://en.wikipedia.org/wiki/File:Mostek_Wiena.svg

[2] Filter Wizard: www.cypress.com/?docID=45637

BK Precision BK560 Programmable IC Tester

The Glory Days of TTL and CMOS

By **Pascal Rondane**
(France)

Like Hewlett Packard, Apple, and Nirvana, the humble beginnings of B&K Precision were in a US garage. In 1948, middle class Americans started buying television sets in numbers. In Chicago a businessman called Carl Korn and his partner Philippe Ban, frustrated by the lack of gear to test television electronic components, decided to create their own test and measurement instruments. Their enterprise was called *Central Television Service Company*. Within a short time, Korn and Ban jointly headed a profitable company selling tube testers and a CRT regenerator. In 1951, the company then named *B&K Precision Corp.* extended its activity into other realms including electronic test and measurement equipment. In 1961 B&K Precision became a subsidiary of a parent company called DynaScan. Sixty years on, B&K are a globally operating manufacturer of advanced test & measurement gear Remarkably, on their website the company name is written as "BK Precision", "B&K Precision Corporation", as well as "BK" in good American tradition to syncopate as much as you can and talk acronyms.

ESTD 2004

Retronics is a monthly section covering vintage electronics including legendary Elektor designs. Contributions, suggestions and requests are welcome; please telegraph editor@elektor.com

Our BK560, my BK560

To get the look & feel of the 1980s two BK560 adverts are reproduced in **Figure 1**. The instrument described here was bought in the year 2000 from a company going out of business I was working for at that time. I also bought a BK502B transistor meter. We had used the BK560 years on end for maintenance work on IBM printers and computers, Motorola and other industrial circuit boards. It was a period the younger generation in this profession will know little about—although these products already contained microcontrollers like the Z80, 8052, and 68000 (and nobody had heard of anything "embedded"), there were also tons of TTL and CMOS logic ICs to grapple with. Our friend the BK560 allowed the team to test an incredible number of these 74xx and 40xx integrated circuits.

Functionality

Two options were available for the user: IC testing "in-circuit" or "out-of-circuit". Incidentally, the BK560 produces so much RF interference VHF radio reception is almost impossible near the equipment—apparently EMC compliance was not a major issue at the time the BK560 was developed and sold.

The version I own contains 1500 components in memory—this is expandable, meaning users can add new components.

The BK560 happily tests counters and flipflops but not monostable (74121, 74123, remember?) with external RC components, or operational amplifiers.

The in-circuit test cannot be performed in circuits where the output is directly connected to the input, which is frequently the case with flipflops, or in circuits connected to a clock source. It is possible though to overcome this problem by disconnecting the clock inputs of the component(s) and using the Learn method described below.

Two modes of operation are available:

1. Out-of-Circuit Testing

On the BK560 keyboard, enter the IC type number. Insert the IC in the socket and launch the test. The display indicates if the IC under test is okay or not. If not, the defective pin number(s) appear.

2. In-Circuit Testing

This can be subdivided into two modes: In-Circuit proper, and Learn.

2a. In-Circuit. You disconnect any clock sources, power the board the suspect IC is on, select the IC type on the BK560, clamp the probe onto the IC pins and start the test.

2b. Learn. Disconnect any clock circuits, power the board concerned, select the IC type on the BK560, and then press the TEST button. The instrument will record and save the result in memory for study and comparison later on. In this mode, when there is a power loss, the information stored in RAM is lost, unless you have wisely installed a bunch of 16 Kbit (2 K x 8) EEPROMs in the sockets provided on the front panel of the instrument (**Figure 2**).

A look inside

Unfortunately I do not have the schematics of the BK560. They can be purchased on the web though for $35 from a BK Authorized Dealer [2].

1

When you open the BK560, you are greeted by a stack of three double-sided circuit boards. The top one accommodates the 20-character fluorescent green display, an 8031-based control section, four type 2764 EPROMs, and one empty IC socket probably intended for software or system extensions. **Figure 3** shows the board stack.

Most of the connectors are fixed with glue dot. Two interesting details on the top board: a length of 16-wire flatcable with connectors at one side only, apparently to avoid complex PCB track routing on the board (**Figure 4**), and an ultra-low-cost PCB corner hinge for the repairman's comfort & delight (**Figure 5**).

On the left, there is another board with a zero insertion force (ZIF) socket for the on-instrument tests, an IDC cable header with eject handles for the test cable, and the four EEPROM sockets.

The power supply comprises a multi-standard power transformer (100–240 VAC, 50 or 60 Hz)

2

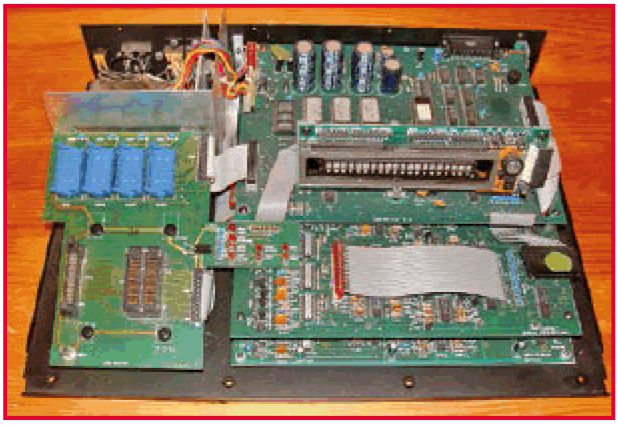
and a couple of linear regulators so typical of at that era.

Testing 1-2-24

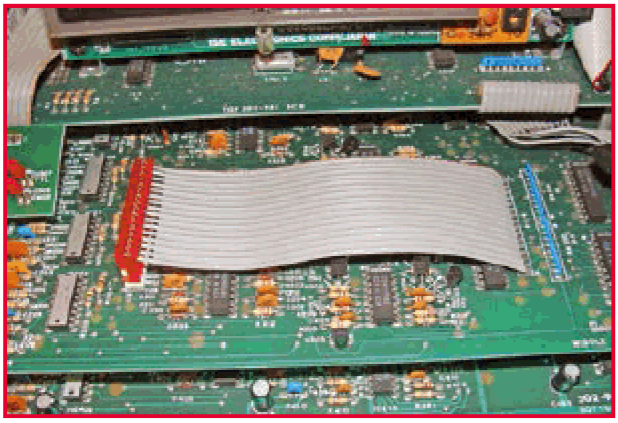
The technical document indicates 24 3-state bidirectional I/O lines available for connecting to pins of the IC under test. The tests on TTL ICs assume a supply voltage, V_{CC}, of 5.0 V, with 2.0 V defined for logic High, and 0.8 V for logic Low. For CMOS ICs, the supply voltage V_{DD} is assumed to be between 5 V and 15 V, with logic High defined as $0.7V_{DD}$, and Logic Low as $0.3V_{DD}$. (Editor's note: "V_{DD}" is the correct notation for supply voltage on CMOS IC logic)

The back panel has a 25-way RS-232 sub-D connector. I can only surmise that the BK560 came optionally with a floppy disk containing the software to support logic ICs not in its resident database. I reckon the optional upgrade should have included a PC connection cable, a software User Manual and an EEPROM for updates. The software User Manual advises the use of an IBM computer or equivalent with at least 250 KB memory; one 5¼-inch floppy disk drive, and DOS 2.00 or higher. For sure we are in the 80's. The user program running on the PC assists with the creation of test procedures for a 'new' or unknown logic IC. Next, the procedures are uploaded to the BK560 and remain permanently in the EEPROM library. This software allows you to define the number of pins and the IC type (TTL or CMOS). In the case of open-collector (OC) outputs, a 10-kΩ load resistor is inserted. A sequential test allows the testing of the individual gates and logic elements in a logic IC (like the 74xx series). Also, a Reset/CLR (RST/CLR) function is available for counters, flipflops, etc. Personally, I don't remember ever having used this function.

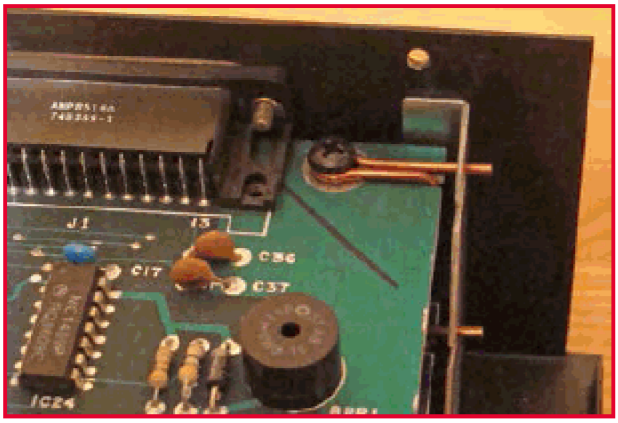
Finally, I was unsuccessful in determining for sure if my BK560 was actually manufactured in the USA, however on first looking it seemed so.



3



4



5

Conclusion

I still use this great piece of 1980's electronic test equipment on occasions for repairs on cards with 74xx TTL or 40xx CMOS series logic ICs. The instrument has given me great service for many years. The only problems I encountered over the years were with the external test cable connector which developed one bad contact. I stumbled on a few more contact problems owing to lacquer or resolderings on the double-sided boards, and some wear and tear inevitably due to over enthusiastic use of the IC probe proper. All things considered, nothing serious or detrimental really.

I have all the original BK560 documents in English and in French but not the schematics. If readers are interested, please do not hesitate to contact me through the Retronics Editor.

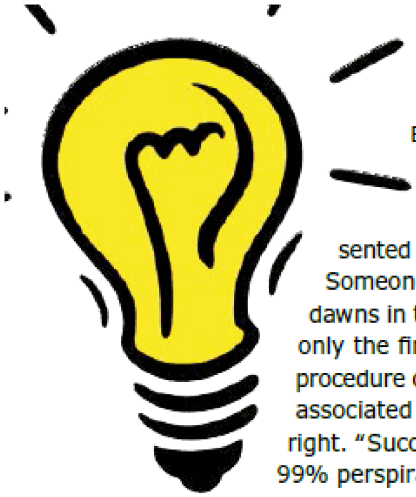
(140051)

Web Links

[1] BK Precision website: www.bkprecision.com

[2] BK560 manuals: www.bkmanuals.com

Engineering Success



By **Gerard Fonte** (USA)

Success is often represented as a light-bulb turning on. Someone sees the light as an idea dawns in their mind. But the idea is only the first step in developing the procedure of success. Edison, forever associated with the light bulb, had it right. "Success is 1% inspiration and 99% perspiration."

Some Assembly Required

While an original concept may be the germ of success, it is rarely the only thing necessary to succeed at something new. It takes a lot of work to make a vision a reality. And until the effort is expended, the vision is only a tantalizing dream that's far away and has no substance. Unfortunately many people think (or perhaps hope) the idea will blossom without exertion. They refuse to nurture and develop their conception. This often requires nurturing and developing themselves.

It's easy to get frustrated when things don't work out as expected. (I know this from vast personal experience.) Buy you learn from this. This method of learning is not something taught in school. They only teach what is already known. Unfortunately they rarely teach how to learn on your own, which seems to be considerably more important. Again, Edison was right when he said "I haven't failed, I just found 10,000 ways that won't work." (On his many attempts to find a suitable material for a lamp filament.) I think most people, including me, would give up well before the 10,000th try. (And, to be accurate, Edison paid researchers to run these tests. He didn't do them himself.) Nevertheless it clearly shows that patience and persistence are sometimes necessary in great quantities.

I have seen too many people give up after a single, half-hearted attempt at something new. "It'll never work." they think. And, quite honestly, with an attitude like that, not working is a very likely outcome. This is because they are giving up on themselves as well as their idea. This illustrates another element of success. You have to have a big ego and have confidence in yourself. If you think, "I can make this work." you probably will.

Big Badda Boom

The Manhattan Project (the US atomic bomb program during WWII) is a breathtaking example of engineering a success. There were uncounted critical elements that were unique in the history of man that all had to work in order for the bomb to explode. (Arguably, the most amazing thing was the production of the man-made element, plutonium, in unbelievably massive amounts.) However, the development of the "explosive implosion" illustrates how success is engineered.

The idea for using explosives to compress a ball of plutonium into a critical mass was completely new. The precision of such an explosion was well beyond anything ever achieved before. (Of course, there wasn't much reason for such precision. Explosives just blew things apart.) Their first attempts were bad. And there was little improvement after a number of further attempts. Oppenheimer called in explosives experts, because they had more practical experience with explosives that the scientists. The experts had problems as well because this was something never before done. However, they had more knowledge than the scientists and were able to better understand the reasons for failure. So they progressed quite quickly and in a relatively short time they succeeded with the explosive implosion.

This illustrates another aspect of success. Sometimes it's necessary to get help. If you are floundering with an idea that you know should work, ask for assistance. No one knows everything and it's not a failure to admit that. I think it's much better to share an idea that works, than be the sole owner of one that doesn't. In this case, you have to set aside your ego. It's difficult. And it does clash with the notion of succeeding by yourself. Nevertheless, teams can accomplish more than individuals. This aspect of success may be the most difficult.

By the Book—NOT!

There are many levels of success. Putting together a kit and having it work is less of a success than if the same person designed the circuit, created the circuit board and assembled it. And different people have different levels of success. But to succeed at something new you have to investigate unexplored ground. You can't just copy something that's already been done. You have "to go where no one has gone before". This is a bit scary but also fun. There's no book to reference. So, you have to do it on your own. It's a lot like learning to ride a bicycle. There will be failures at first. No one just gets on a bike and starts riding. And it will take some time to learn from these mistakes. But, eventually, riding a bicycle becomes a simple task. The skinned knees and bruises are forgotten. And when you learn to ride a bicycle, you find your horizons have expanded. You can travel farther and faster than ever before. The tools of creating success are personal. This means that anyone can learn them. It does take some effort. But any new endeavor does. And the more you use these tools, the better you will become. Success breeds success. After a while it becomes a habit—just like riding a bicycle. And when it becomes a habit, you will expect to succeed rather then hope to succeed. Success is a manufactured thing.

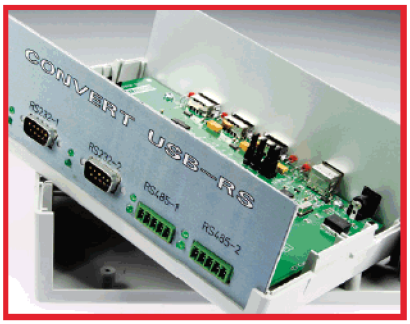
(140299)

•Regulars

NEXT MONTH IN ELEKTOR MAGAZINE

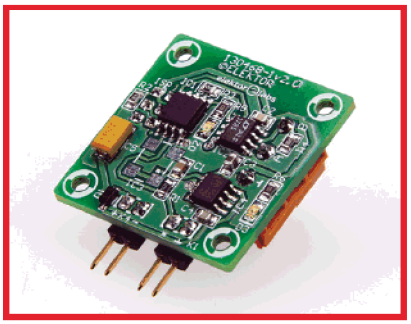
Nostalgic Nixie Clock

For some reason Nixie tubes continue to exert a great attraction to electronics folks. Various projects based on these tubes got published in Elektor already and to these we now add a simple Nixie Clock with a twist: it receives time information via a built-in GPS module. That's a nice combination of old and new electronics!



USB Hub with Legacy RS232 and RS422/485

Electronics designers often run into a problem with modern computers no longer having the legacy serial interfaces, while many microcontroller-circuits rely on them for communication. This handy circuit offers a universal solution: it contains a USB hub with three USB connections, and in addition has two full duplex RS232 and two RS422/485 ports.



Sensor Board for ElektorBus

The RS485 bus is perfect for remote reading of temperature sensors across large distances. This compact sensor board is equipped with an ATtiny microcontroller and an RS485 driver. Up to four sensors can be connected. The associated firmware uses the ElektorBus protocol for data transfer, and demo PC software is also available.

Article titles and magazine contents subject to change, please check www.elektor-magazine.com for updates.
Elektor's October 2014 edition is processed for mailing to US, UK and ROW Members starting September 11, 2014.

Please note: as of the October 2014 edition Elektor magazine is no longer available from bookshops, newsstands and kiosks. Readers not having an Elektor membership can purchase printed or digital copies of individual magazines directly from the publishers at www.elektor.com (click on MAGAZINES).