elektor electronics
leading the way

September 2004
£3.70

www.elektor-electronics.co.uk

plus: Buyers Guide

Bluetooth Remote Control

USB Host Controller

TRENDS in MICROCONTROLLERS

step-by-step construction: Rolling Dice

...control, measure, drive and switch...

Swiss Army Knife

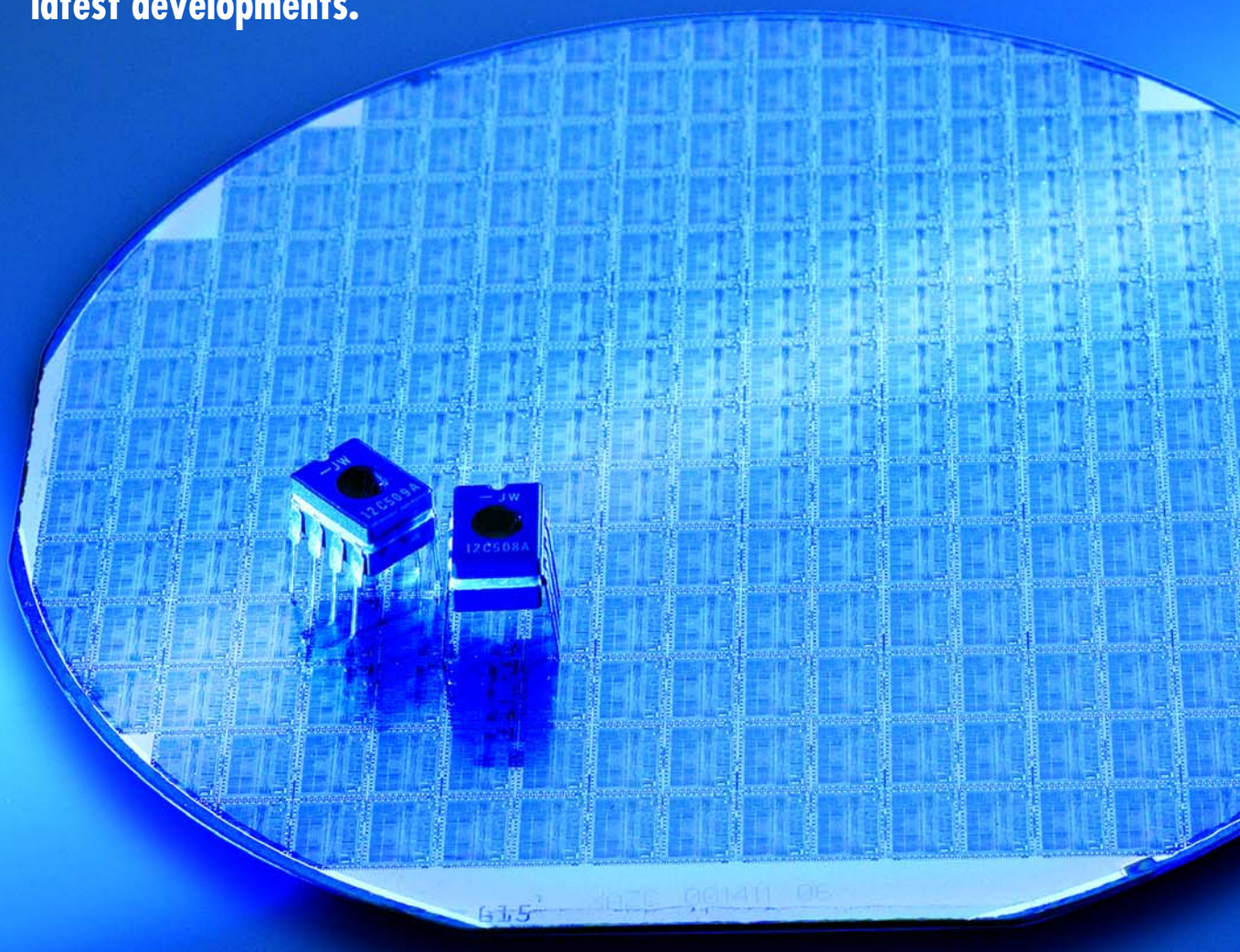
09

9 770268 451111

TRENDS IN MICROC

Christian Tavernier

The number of different microcontrollers on offer just keeps on growing. Nearly every single week a new product appears on the market. Choosing the right microcontroller for a particular design is becoming increasingly more difficult. Reason enough to appraise the latest developments.



ONTROLLERS

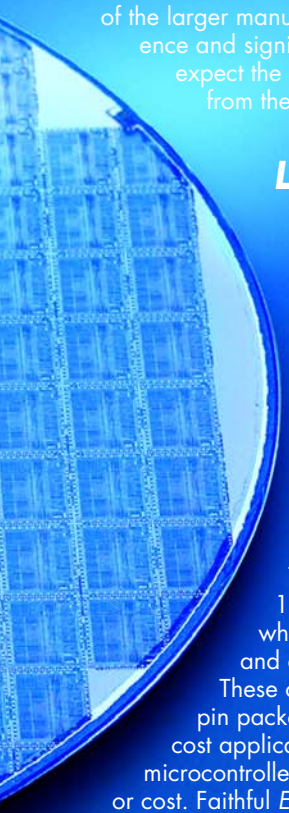
An abundance of choice

The microcontroller market is mostly dominated by a few 'heavyweights' such as Microchip, Atmel and, to a lesser extend, Philips. In addition, there are many smaller manufacturers who make very interesting developments in this area. There are also a number of manufacturers who are not at all that well known for their microcontroller products, and are often forgotten as a consequence. Names that come to mind are Toshiba with its TMP86xxx-family, and Zilog, the inventor of the famous Z80, who now offers the Z8 Encore! and eZ80 families. Also don't forget Dallas Semiconductor with its DS89C420, Cypress with its PsoC (Programmable System-on-Chip) such as the CY8C27x, or the, until recently, completely unknown company Cygnal with its C8051xxx.

16-bit devices appear to be old hat

The majority of these products, however interesting, will have to have some very desirable properties in order to secure a prominent spot in an already oversupplied market. A few years ago, the PIC from Microchip succeeded in doing this. For other manufacturers this is not a problem; Toshiba, for instance, makes enough end products such as notebooks and other consumer electronics to use plenty of its own microcontrollers.

In this article we will concentrate mostly on the products of the larger manufacturers, because their broad experience and significant head start means that we can expect the most innovative developments to come from them.



Larger or smaller?

A very obvious trend is that the microcontrollers are becoming increasingly more powerful: more memory, more inputs and outputs, more integrated peripherals, etc. A lesser-known development is currently in progress at the bottom end of the market, both at Microchip, and, to a smaller extend, at Atmel.

For example, Microchip developed the 12Fxxx from the 12Cxxx. The different letter indicates an important difference: the 12C is an OTP (One Time Programmable) device, while the 12F is provided with Flash memory, which can be electronically programmed and erased a few thousand times.

These chips are available in either 8- or 14-pin packages and are mainly intended for low cost applications where until a few years ago microcontrollers were avoided because of their size or cost. Faithful *Elektor Electronics* readers will know the 12C508 (**Figure 1**), the first member of this family. But let us first take a look at the 12F629, which, in its 8-pin package has the following to offer: 1 k Flash program memory, an EEPROM 128 bytes in size, 64 bytes of RAM, 6 I/O-lines, an 8-bit and a 16-bit programmable timer. If all of this is not enough, the 12F675 may be more appropriate. Incidentally, the numbering scheme escapes us. This one is identical to the 629, but in addition provides a 10-bit, 4-channel ADC. These ICs operate from an internal RC clock source up to 4 MHz or from an external crystal up to 20 MHz.

Atmel hasn't been sitting idle in this area either, and has the ATtiny-family, which comprises nine different versions. Even though it can't really be called new, the ATtiny2313 deserves some of our attention. It has a 2 k Flash memory, a UART (Universal Asynchronous Receiver/Transmitter) and up to 18 I/O-lines. The crystal frequency may be up to 20 MHz. The controller is housed in a 20-pin package.

The rfPIC or 'radio' PIC

In the area of 'small' microcontrollers it is Microchip who steals the show.

Now that microcontrollers are becoming more common in remote controls, Microchip has decided to offer the rfPIC, literally the radio frequency PIC. This one is part of the small PIC-family that we discussed just before. In a single package, the rfPIC contains, in addition to the microcontroller, a complete circuit for an ASK- or FSK-transmitter (Amplitude or Frequency Shift Keying). Also provided, to ensure a stable frequency, a VCO (Voltage Controlled Oscillator) and a PLL (Phase Locked Loop), based on the crystal frequency of the circuit.

The basic version, the rf12C509 comprises a 12C509 and a built-in transmitter. But for the creative among us the rf12F675 is much more interesting. This is, after all, the aforementioned 12F675 with the radio added. Applying these controllers is very simple, as is illustrated by **Figure 2**.

Trends

Beside the usual increase of on-chip, integrated memory, the last few years show two more main areas of microcontroller development.

The first concerns the operating speed. Although countless circuits are still provided with a 4-MHz crystal, the product catalogues from the Microchip, Atmel and Motorola are full with ICs that will function at 16 or 20 MHz. Note that it is not that straightforward to just compare clock speeds. An ATmega128 from Atmel operates on a clock frequency of 'only' 16 MHz, but executes nearly all instructions in a single clock cycle. On the other hand, an MC9S12D from the HCS12 family from Motorola runs at 25 MHz but requires multiple clock cycles per instruction.

The second main trend concerns the specialisation of the internal peripherals. The UART and SPI interfaces have been common in the higher-end microcontrollers from most manufacturers for many years. But now more specialised interfaces are being added. Virtually all manu-

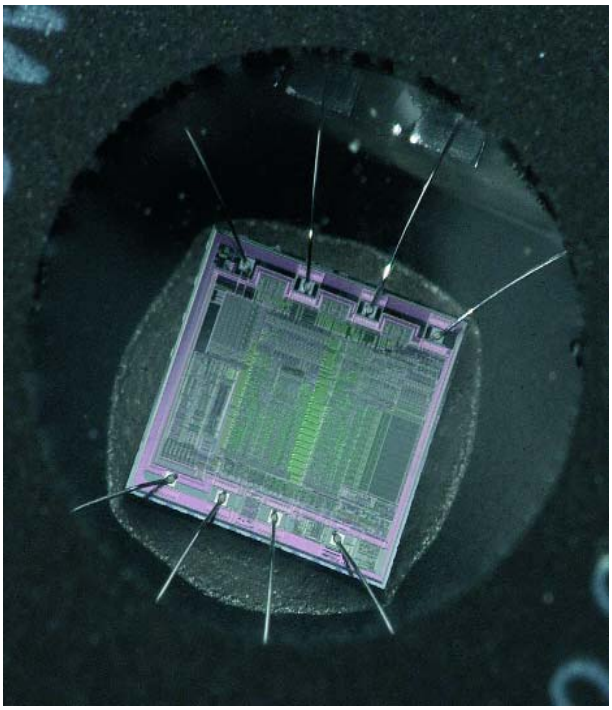


Figure 1. A window through which the memory is erased with ultraviolet radiation. This image will soon be condemned to the past as a consequence of the rise of Flash memory.

facturers support the I²C bus. Sometimes the interface operates only in the simplest slave mode, but more often than not, in master mode as well. The USB (Universal Serial Bus) interface, popular because of the PC-market, is quickly making its entry into the microcontroller world. Microchip has the 16C745, the 765 and the future 18F2455, 255, etc. Atmel offers the AT91RM3400 and even Motorola has adapted a version of the old 6805, such as the 68HC705JBx.

ARM

On our whirlwind tour of the microcontroller world we cannot ignore the products from ARM. Although ARM, strictly speaking, doesn't actually manufacture microcontrollers, it has provided for many years the 32-bit cores that well known manufacturers such as AMI, Atmel, Cirrus, Philips, Samsung, STMicroelectronics, Texas Instruments and even Intel happily use in their own products. We get the impression that this concerns an important development. Why worry about developing your own 32-bit microcontroller when an industry-accepted core already exists? The need for increasingly advanced products, which, at the same time, are still easy to maintain and use, inevitably influences the intelligence of each system: the microcontroller. The need for an ever-increasing number of functions translates into a transition from the traditional 4- or 8-bit controllers to 32-bits. It seems that progress has skipped over the 16-bit versions, because 32-bit offers a much higher performance and unequalled flexibility.

Applications

Especially in cars, the use of microcontrollers is becoming increasingly important. This has resulted in manufacturers not only integrating CAN-bus controllers on their chips but the microcontrollers themselves are becoming more advanced. This is the case, for example, with the 18F2332 from Microchip, which can operate at 40 MHz. The same is true for the MC68HC908MRx from Motorola.

The use of microcontrollers in cars is becoming increasingly important

Even in this ocean of digital chips, the analogue world has not been forgotten. These days, all manufacturers have ICs in their product catalogues with advanced analogue/digital-converters. An example is the ATmega128 from Atmel. It contains a 10-bit, 8-channel model, which operates in normal mode (single ended) as an 8-channel converter. 7-channel differential mode is also possible and 2 channels are provided with an integrated programmable amplifier (1 to 200 times). For many applications there is no longer the need to resort to an external converter.
The addition of internal peripherals has no negative impact on the amount of memory. On the contrary, the better-endowed ICs often contain memories of amazing size. **Table 1** provides an overview of the three 'largest' presently available microcontrollers from Microchip (the 18Cxxx-family), from Atmel (the ATmega-family) and from Motorola (the HCS12-family). Conspicuous in this table is the MCS12H256B from Motorola (**Figure 3**) with the overwhelming 99 parallel I/O-lines and 256K-Flash memory. This is quite a bit different than the 68705P3 with 20 I/O-lines and 1.6K-memory.

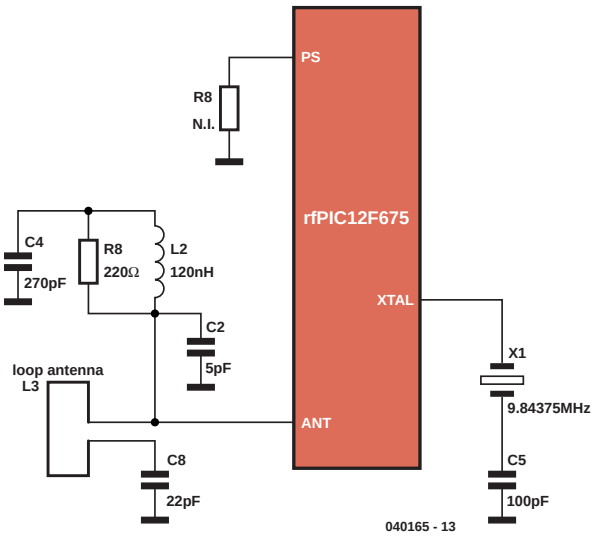


Figure 2. The RF part in an rfPIC is easy to use.

Phoenix

You know, the mythical bird that rose from its own ashes. There is a microcontroller that we certainly have to give some attention despite the fact that we're now in the second half of 2004: the 8051. Even though this microcon-

troller, in its basic form, first appeared on the market at the end of the 80's, it is still the basis for many new developments.
Despite its respectable age, the 8051 has survived thanks not only to the addition of new internal peripherals and expansions to the program memory, but also additional features that used to be considered impossible (such as in-circuit-programming, discussed further on in this article).

If the 8051 assembler doesn't hold any secrets for you any more, then you should take a look at the AT89xxx family from Atmel. All ICs in this family were developed from the 8051, but have in-circuit-programming, new internal peripherals and expanded program memories. If you find that these ICs are a little too 'well-done' then maybe the new Philips family P89LPC900 has something to offer you. If this name doesn't mean anything to you then you should dive into your *Elektor Electronics* archive for the October and November 2003 issues in which this IC was discussed extensively.

Programs and development tools

Initially, that is to say, when a microcontroller barely contained 1 or 2 kB of memory, programming in assembler was the only real option. The available memory would easily have been filled with a few lines of a higher programming language, because a single line usually results in numerous lines of assembly language as a result of the 'expansion urge' of the compiler.

The assembler is still an indispensable tool if we want to create a genuine real-time program or if we want to obtain the greatest possible performance out of the microcontroller. But these days, the preference is to use a higher programming language. Most programmers now select either BASIC or C in preference to assembly. There are now C compilers for all microcontroller families. This varies from the quite pricey products such as the

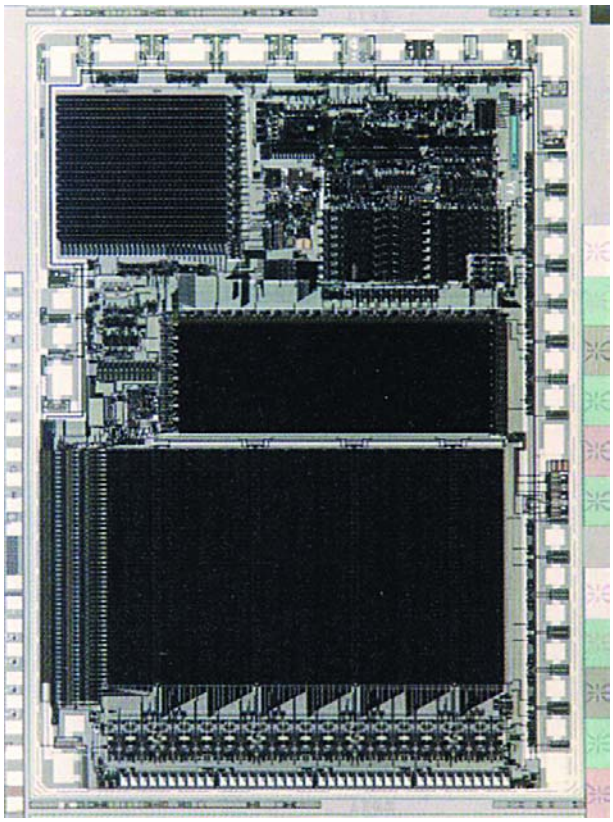


Figure 3. Photograph of a microcontroller-chip.

Table 1. Main specs of three modern controllers from Atmel, Microchip and Motorola.

Parameter	ATmega128 (Atmel)	18F8720 (Microchip)	MC9S12H256 (Motorola)
Flash program memory	128 K	128 K	256 K
EEPROM data memory	4 K	1 K	4 K
RAM memory	4 K	3.8 K	12 K
8-bit timers	2	2	0
16-bit timers	2	3	1
Capture and compare timers	4	5	8
PWM channels	8	5	6
I/O lines	53	68	99
UART (serial asynchronous interface)	2	2	2
SPI (serial synchronous interface)	1	1	1
I²C interface	–	1	1
A/D converter	8 channels	16 channels	16 channels
A/D resolution	10 bits	10 bits	10 bits
Analogue comparators	1	2	-
Maximum clock frequency	16 MHz	40 MHz	32 MHz

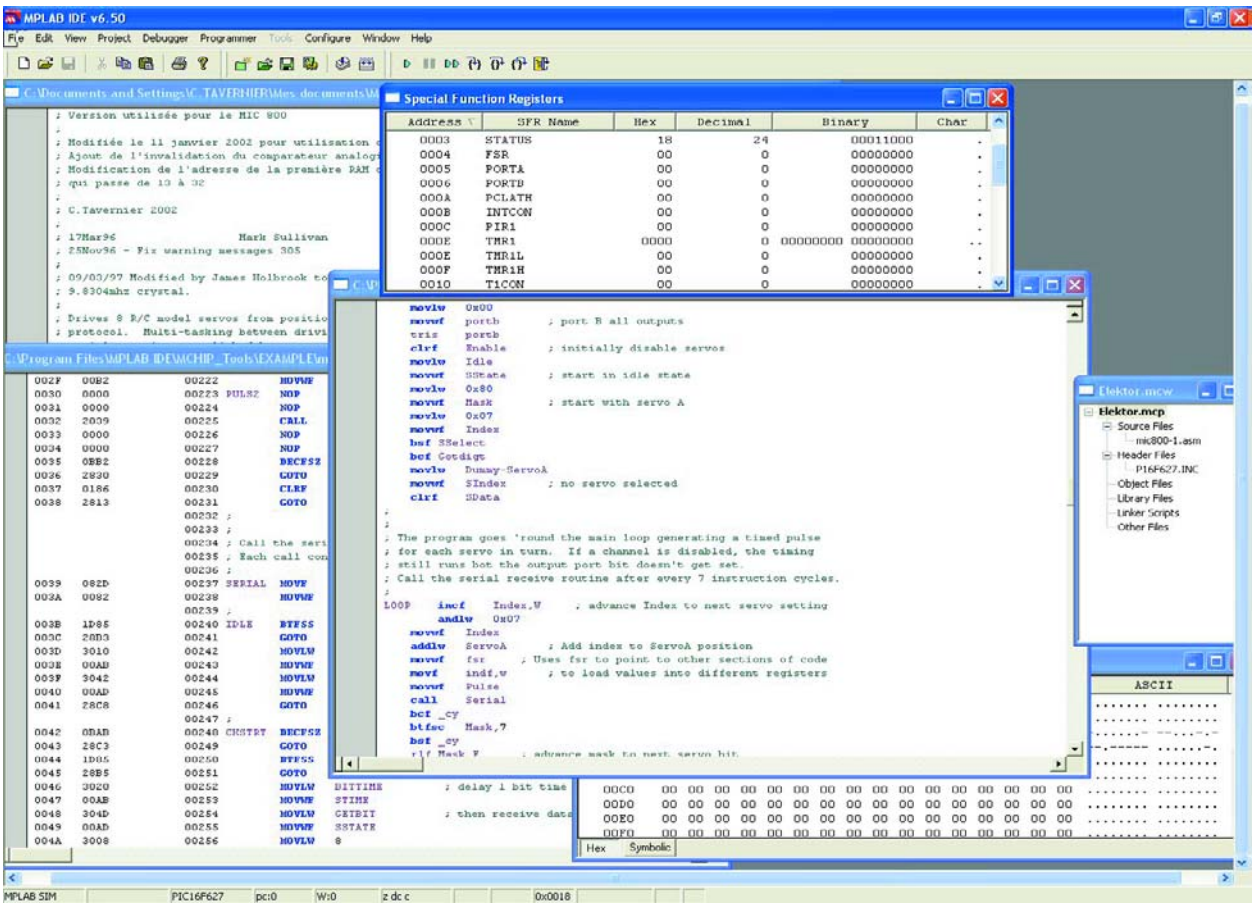


Figure 4. Although MPLAB is free, it contains everything to develop an application for a PIC with a maximum of user friendliness.

CCS C PIC compiler to the freeware PICC-Lite C compiler from Htsoft, which is often sufficient for many applications.
For those who prefer not to work in C or assembler there are excellent BASIC-compilers, such as the PicBasic Pro Compiler from Micro Engineering Labs and Proton from Crownhill Associates. For the AVR-family from Atmel you could use Bascom AVR from MCS Electronics.

Most programmers now prefer BASIC or C over assembler

For the 8051 family there is, among others, a special Bascom version from MCS, and SDCC, a free C compiler. Only the Motorola products suffer from a sparse availability of compilers, since the amount on offer is absolutely minimal.
Accepting the fact that most software developers often use identical programming elements, Microchip has come up with an original idea: a new program called Maestro. Using this, you can develop an application with next to no actual programming. This development environment can be downloaded free from the Microchip website and makes it possible to stick together standard modules. In this way a program can be put together nearly automatically.

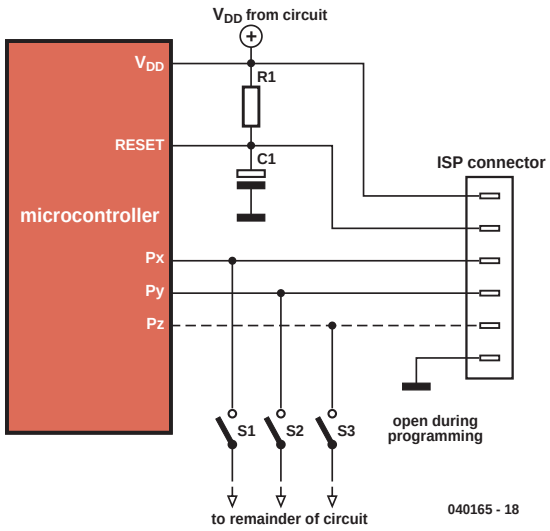


Figure 5. Operating principle of in-circuit programming (ISP or ICSP).

In addition to a compiler, a microcontroller project also requires other development tools. Microchip and Atmel have the upper hand here with MPLAB (**Figure 4**) and AVR Studio respectively. These development environments, running under Windows, contain everything a programmer could only dream of a few years ago, including a program simulator. This development is still in progress, because MPLAB is currently up to version 6.5 and AVR Studio is up to version 4, so that even new products are well supported.
Despite the remarkable number of features these development tools are completely free. You only need to download them from the manufacturer's website to get started. It has to be noted that Motorola and other lesser-known

manufacturers are running behind in this area, certainly when compared with these free products.

In circuit programming

Although it isn't really new, ISP (In System Programming) or ICSP (In Circuit Serial Programming) is without doubt the most important development regarding the programming aspect. When the programmer has created the program in the correct form, microcontrollers that support it can be programmed from a standard PC via a very simple interface, while the controller itself can remain in the circuit.

Figure 5 illustrates the operating principle of this concept. The microcontroller obtains its power supply and clock signal from the PCB. Two or three port pins on the IC temporarily have an alternative function, which makes it possible to erase and program the program memory. If the application doesn't use these pins, then they may be connected directly to a PC. Otherwise a few jumpers or DIP switches may be required to isolate the circuit during programming.

All modern microcontrollers with Flash memory support this programming method and on the Internet countless free software can be found for all the various controllers to program them in this way.

A program that distinguishes itself in this respect is FLIP (**Figure 6**). It is offered free by Atmel and can program countless microcontrollers from this manufacturer, provided they have some kind of serial interface (RS232, SPI, USB and even CAN-bus). This is currently the most versatile program around.

Microcontrollers or not?

In this article we must not forget to mention special microcontrollers such as the Basic Stamp, the PIC Basic and the Basic Tiger. These ICs with 24, 28 or 40 pins have, on a tiny PCB, a fast microcontroller, which is pre-programmed with an interpreter for a higher language. This is usually BASIC, but other programming languages are emerging, such as Java for the Javelin Stamp from Parallax. Even programming in an object-oriented language is possible with the OOPic.

Even though these products are very successful, they are very expensive compared to real microcontrollers, mainly because of they way they are manufactured. They seem to be most suitable for experimental applications or small production runs.

Conclusion

Much more can be said on this topic. But if we have to summarise the developments of the microcontroller area in a few words, then we can say that the advance of microcontrollers in all aspects of electronics is becoming increasingly important. These can be either more powerful devices on the one hand or with ever smaller and cheaper versions on the other hand. The application of microcontrollers is becoming easier and easier with free development tools and powerful compilers.

The microcontroller has become an indispensable part of electronics. This is testified by the many projects containing microcontrollers that have already appeared and will appear in *Elektor Electronics*.

(040165-1)

> Free development tools running under Windows contain everything that programmers until recently could only dream of

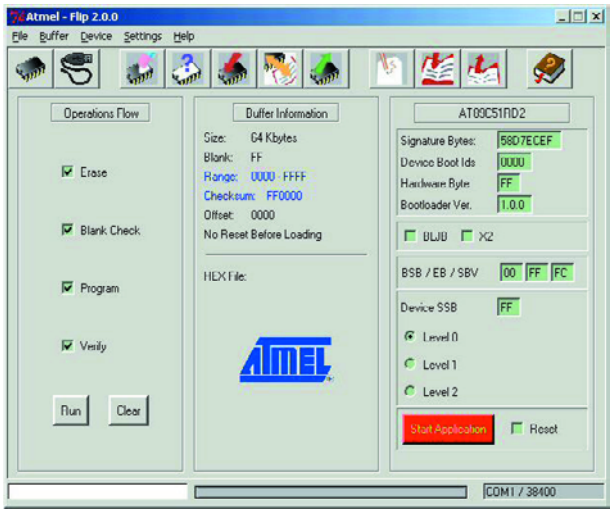


Figure 6. The FLIP software program from Atmel for the in-circuit programming via practically any serial connection.

Internet addresses

Author's website (in French): http://www.tavernier-c.com/

Conventional microcontrollers and development tools

ARM:	http://www.arm.com/
Atmel:	http://www.atmel.com/
Microchip:	http://www.microchip.com/
Cygnal:	http://www.cygnal.com/
Cypress:	http://www.cypress.com/
Motorola:	http://mot-sps.com/
Philips:	http://www.semiconductors.philips.com/
Toshiba:	http://www.toshiba.com/taec/
Zilog:	http://www.zilog.com/

Special microcontrollers

Parallax:	http://www.parallax.com/
OOPic:	http://www.oopic.com/
Basic Tiger:	http://www.wilke.de/

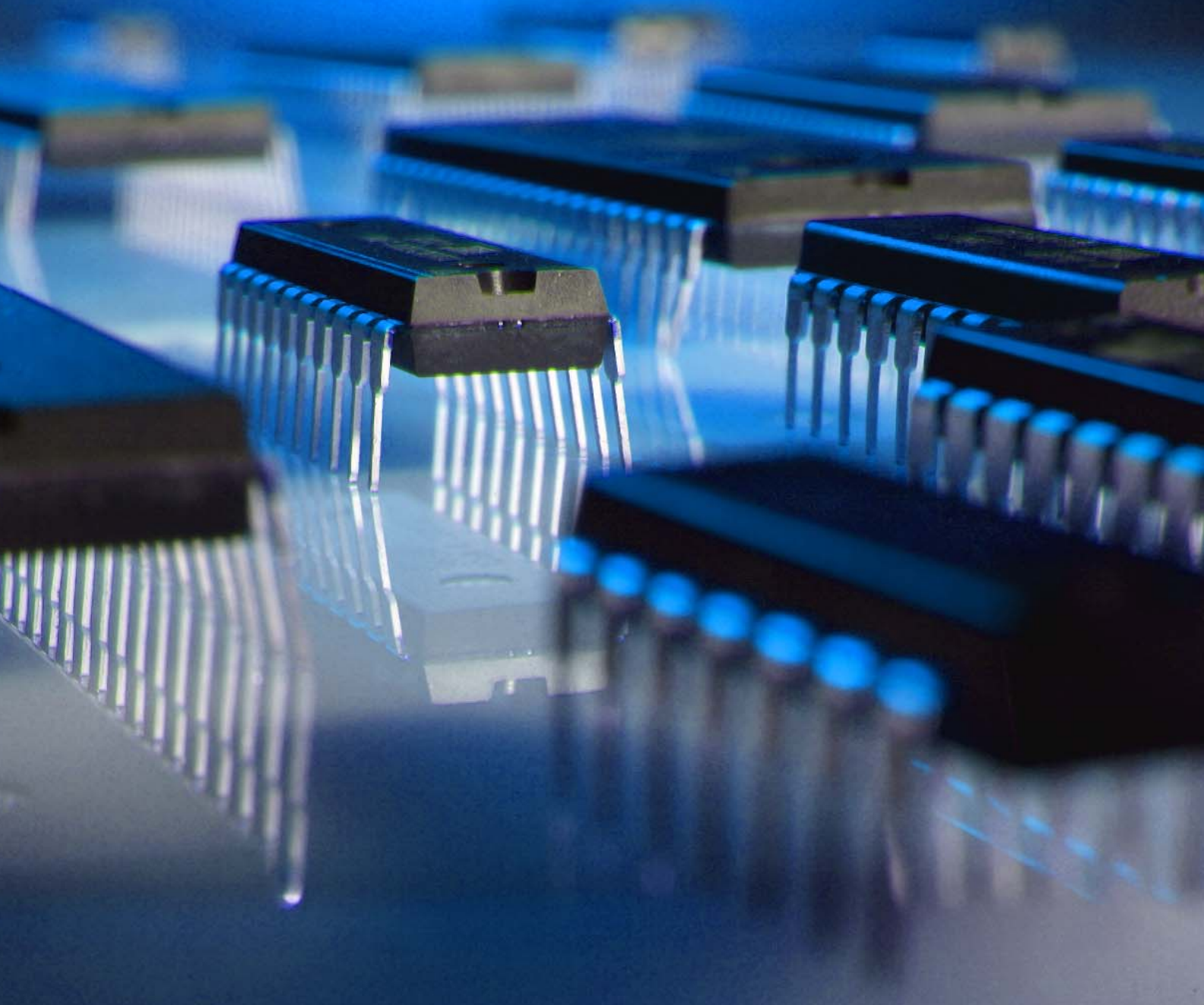
Compilers

CCS C Pic Compiler:	http://www.ccsinfo.com/
PICC-Lite C Compiler:	http://www.htsoft.com/
Bascom AVR en Bascom 8051:	http://www.mcselec.com/
Proton - Basic compiler:	http://www.picbasic.org/
PicBasic (Pro) Compiler:	http://www.melabs.com/
SDCC Compiler:	http://sourceforge.net/projects/sdcc

MICROCONTROLLER

A dazzling choice of devices...

Lots of semiconductor manufacturers also supply microcontrollers. The plethora of available versions and their increasing flexibility do not make it easy to find the right micro for your specific application.



BUYERS GUIDE

Comparing and weighting microcontroller specifications being largely subjective, it's just not possible to give hard and fast rules to anyone wishing to pick 'the best' microcontroller for a given job. In nearly all cases the starting point will be the actual application with its very specific requirements. This article does not aim at comprehensiveness by listing thousands of different microcontrollers but rather summarizes selection criteria to help you reduce the number of micros you could use from stunning to manageable.

Speed
What is commonly referred to as the speed of a microcontroller is not just dependent on the maximum CPU clock and the clock generator (quartz crystal) — you also need to look at the number of clock cycles the micro takes to execute an instruction, as well as the programming language used (assembler may be many times faster than a higher language). Depending on the clock frequency, universal controllers are suitable for applications well into the megahertz range (video processing).

Program memory
The program to be executed by the micro is stored in non-volatile memory. An internal OTP EPROM can be loaded once only, hence the more expensive Flash version of the controller is usually employed during the program development phase. Flash memory can be loaded in seconds and is equally simple to erase. This may be done using a programmer or in-circuit (ISP). These days the size of conventional internal program memory ranges from zero to 1024 kB of Flash memory (perhaps even more?).

Microcontrollers with a little window for EPROM erasure using UV light are now old hat. External windowed EPROMs are now only used in the case of very large programs — Flash RAMs are increasingly seen instead of EPROMs.

EEPROM
When program variables are to be retained even if the microcontroller is switched off completely, EEPROM non-volatile memory, internal or external, is called for. Contrary to some popular beliefs, the number of write operations sustained by EEPROMs is **not** infinite.
In general, external EEPROMs are connected to the microcontroller via a two-wire bus. They are typically used when certain application-specific data (like calibration values) are to be read at the start of the program.

RAM
RAM is used to store variables during program execution. RAM on board microcontrollers is usually limited to 4 kB and the actual requirement is often much lower. External RAM is also possible.

Digital Input/Output (I/O)
The number of digital I/O lines you'll need in your target application should be easy to tell. However, if the internal resources are exhausted, I/O may also be used to connect peripheral circuits. For simple applications you have a wide choice of controllers sporting little I/O (for example, Atmel's 'Tiny' devices with just eight pins). In some cases, whole ports are required in parallel mode, and that's when you cannot avoid those difficult to solder multi-legged beasts.

Timers/counters
If a program is to measure periods, or count events, then the controller should have timers and/or counters on board. Fortunately, most current models contain up to three each of 8-bit or 16-bit timers/counters under the control of internal registers. Timers/counters are also needed to generate a clock-independent signal (PWM, UART).

The Watchdog timer is a special case. It is set to an interval by the running program having to reset it all the time.

External interrupts
Not only timers/counters generate interrupts. When an external event is to halt the main program execution and force a service subroutine to be run, one or two external interrupt inputs are available on most micros for this purpose.

Interfaces
Very useful to have are on-chip industry standard interfaces like I2C, I2S, SPI, CAN, USB, LIN or one for a common LCD. True, such interfaces can be emulated in software, but you'll find that doing so takes time and deep knowledge of assembly code programming.

Analogue hardware
Microcontrollers frequently feature integrated interfaces to the analogue world. These interfaces include analogue/digital converters (with different resolutions and analogue multiplexers in front of them), analogue comparators and even operational amplifiers (with output to a pin).

Modes of operation
These are interesting to look at if a micro is to work in a battery-powered circuit. A number of sub-circuits in the controller may be switched to 'sleep' mode to save power.

Programmability
When the controller has an ISP interface, it can be programmed in the host circuit. If not, you'll need a more or less complex programmer. An ISP-less controller soldered into a circuit can not be reprogrammed without a lot of work.

Special features
There exist microcontrollers that are only suitable for a specific application like motor control, DSP and controllers with an RF input section. If you're planning a related application, these special devices may be well worth considering.

Price, availability, case
The price of a microcontroller is not too important if you are doing a one-off project or a small series. However, there's little satisfaction in finally having found the ideal type for your application and then discovering that the chip is only available in 10,000+ quantities directly from Korea. The enclosure your 'dream' micro comes in is also an important factor, after all, who's capable of manually soldering a 256-pin 'flat-something' case with pins at sub-millimetre distance?

Development Systems & Co.
This point is of marked interest to semi-professional developers. If you program in assembler, every new controller family you embrace requires a new language to be learned. If you do not like that, you either stick to the same controllers for years or invest in a higher-language compiler like C, Pascal or Basic. These products are usually suitable for several controllers, but tend to generate bulkier code than assembler.
To this should be added the cost of a development system. For some controller families, IDE's are offered at no cost while for others you need to dig deep in your pocket. In particular commercial programmers that come with complex 'pods' (controller sockets) may have extortionate price tags. (040286-1)

Microcontroller manufacturers overview

4-, 8-, 16-, 32- and 64-bit families

Altera - www.altera.com
32-bits: EPXAxxx (ARM V4T), Nios (Nios)

Analog Devices - www.analog.com
8-bits: ADuC8xx (8051)

AMD - www.amd.com
32-bits: Au1x00 (MIPS)

Atmel - www.atmel.com
4-bits: T48C510, ATAxxx (MARC-4)
8-bits: AVR (AVR), AT89xxxx (8051), Mega AVR (AVR)
16-bits: C251 (8051), AT91xxxx (ARM)

ARC International - www.arc.com
32-bits: ARC501 (ARCompact), ARC7xx, ARC6xx, ARCtangent (RISC)

ARM - www.arm.com
32-bits: ARM10xx, ARM11xx, ARM7xx, ARM9xx, SCxxx, MPCore (ARM)

Cirrus Logic - www.cirrus.com
32-bits: CS89712, EP73xx, EP93xx, PS7500xx (ARM)

Cybernetic Micro Systems - www.controlchips.com
8-bits: P-51 (8051)

Cygnal Integrated Products - www.cygnal.com
8-bits: C8051Fxxx (8051)

Cypress Microsystems - www.cypressmicro.com
8-bits: CY8C2xxxx (M8C)

Dallas Semiconductor (Maxim Integrated Products) - www.maxim-ic.com
8-bits: DS2xxx, DS5xxx, DS80Cxxx, DS87Cxxx, DS89Cxxx, MAX765x (8051)

Fujitsu Microelectronics - www.fujitsu.com
8-bits: MB89xxx (F2MC-8L)
16-bits: MB90xxx (F2MC-16)
32-bits: MB91xxx (FR)

IDT - www.idt.com
32-bits: RC32xxx (MIPS)

Infineon Technologies - www.infineon.com
8-bits: C5xx, C868 (8051)
16-bits: C16xxx (C166 v1), XC16xxx (C166 v2)
32-bits: TC11IB, TC19xx (TriCore V1.3), TC17xx (TriCore V1.2)

Intel - www.intel.com
8-bits: 8xC251x, 8xC51xx (MCS51)

16-bits: 80C18x, 8xCx96xx (=MCS-x96xxx)
32-bits: 80960 (i960), IXC1100, IXP4 (StrongARM v5TE)

Microchip Technology - www.microchip.com
8-bits: PIC12xxx, PIC14xxx, PIC16xxx, PIC17xxx, PIC18xxx, rfPIC (PIC micro)
16-bits: dsPICxxxx (Modified Harvard RISC)

MIPS Technologies - www.mips.com
32-bits: 4Kxx , M4K, 24Kx (MIPS)
64-bits: 20k, 10Kx, 5Kx (MIPS)

Motorola Semiconductor - www.freescale.com
8-bits: MC68Hxxx (HC0x)
16-bits: HCS12x, M68HCxx (HCS12)
32-bits: 68300 (68K), MCF5xxx (ColdFire), Mcore xxx (RISC), MAC7xxx (ARM), MPC5xxx (PowerPC)

National Semiconductor - www.national.com
8-bits: COP8xxxx (Modified Harvard RISC)
16-bits: CR16xxxx (Compact RISC)

NEC Electronics - www.necel.com
8-bits: 78K0S/Kx1, 78K0/Kx1 (NEC K)
32-bits: v850ES/Kx1 (v800)
64-bits: VR41xx, VR5xxx (MIPS)

Oki Semiconductor - www.okisemi.com/us
4-bits: MSM6318xx, MSM6415xx (nX)
32-bits: ML67xxxx (ARM7TDMI)

Philips Semiconductors - www.semiconductors.philips.com
8-bits: P8xC5x, P89LPC9xx, P8xLPC76x (8051)
16-bits: PXAxxx (XA)
32-bits: LPC2x0x (ARM7)

Rabbit Semiconductor - www.rabbitsemiconductor.com
8-bits: Rabbit 2000, Rabbit 3000 (Z80/180)

Renesas Technology - www.renesas.com
4-bits: M45xx (720), H4xxx (HMCS400)
8-bits: M38xx, M78xx, M3754x, H8/380xx (H8)
16-bits: M77xx, M79xx (740), H8/30xxx (H8), H8S2xxx (H8S), M16C/xx, M32C/xx (M16C), H8/36xx, H8SX/1xxx (H8)
32-bits: SH-xxxx (SuperH), M321xx (RISC)

Silicon Storage Technology - www.sst.com
8-bits: SST89xxxx (FlashFlex 51)

Sharp Microelectronics - www.sharpsma.com
16-bits: LH754xx (ARM)
32-bits: LH7952x, LH7A4xx (ARM)

STMicroelectronics - www.stm.com
8-bits: ST62xx (ST6), ST72xxx, ST7FLite (ST7), uPSD3xxxx (8032)
16-bits: ST10xxxx (80C166), ST92Fxxx (ST9), STR7xxx (ARM)
32-bits: ST40RA (SH4)

SuperH - www.superh.com
32-bits: SH-4xxx (SuperH)
64-bits: SH-5xxx (SuperH)

Tensilica - www.tensilica.com
32-bits: Xtensa V, Xtensa LX (Xtensa)

Texas Instruments - www.ti.com
16-bits: MSP430xxxx (MSP), TMS470 (ARM)

Toshiba America Electronic Components - http://chips.toshiba.com
8-bits: TMPx8xxxx (TLCS)
16-bits: TMP96xxxx, TMP91xxxx, TMP95xxxx, TMP93xxxx (TLCS)
32-bits: TMP92xxxx, TMP94xxxx (TLCS), TMPR19xxx, TMPR39xxx (MIPS)
64-bits: TMPR49xx, TMPR99xx (MIPS)

Triscend - www.triscend.com
8-bits: E5 (8051)

Ubicom - www.ubicom.com
8-bits: SXxxxx, IP2012/2022 (MASI)
32-bits: IP3023 (MASI V2)

Xemics - www.xemics.com
8-bits: E88LC0x (RISC)

Xilinx - www.xilinx.com
32-bits: PowerPC 405 (PowerPC)

Zilog - www.zilog.com
8-bits: eZ80xxx, Z8xxxx, Z8Fxxx, Z8 Encore! (Z80/180)

Swiss Army

Jim Spence

In a fix? Need to get a microcontroller project off the ground FAST? This article proves that an ultra-versatile microcontroller board can be built and programmed even by relative newcomers. The main circuit is based on the Atmel 89C8252 which has an 8052 architecture.

Knife

Tiny BASIC, 8051 assembler, RS232 and USB all in one project

Included is some very special (free) software that enables even the most code shy among you to make the controller do something. There are two alternatives to communicating with the circuit: USB 2.0 or RS232.

The original intention of the project was to produce a controller with built in BASIC that could retain the program after switch off, automatically start at switch on and not consist of too many chips. It should also be very easy to use and not require any special software on the PC. This has been achieved with more or less a single chip. This is possible because the 89C8252 has 2 kB of data EEPROM that can be programmed with high level instructions. The AT89C8252 micro was first used in the Elektor *'51 Flash Micro Board* published in December 2001 and now a 'classic' with PCB sales in the 'k' range.
You may well be wondering why not use a BASIC or C compiler on your PC and blow the object code into the micro. These are good options, however you loose the immediacy of your actions. The Swiss Army Knife board *immediately* executes any code sent to it. Typing:

```
Pz1=0
```

at the console will immediately set all of the port 1 lines to Low. Also, a level of knowledge is required of the BASIC and C compilers before anything can be achieved. If you simply want to get on with your favourite robot project, then this is the way to do it. Mind you, you can still use the compilers or assembler later when you have done some experimenting.

RS232 and/or USB

Along the way the author got fed up with incorporating a dedicated communication system with every board especially when it was not usually needed in the final application. This lead to a separate power supply and RS232 connectivity that could be used with other circuits. However, as the RS232 format is getting a bit rusty and is not even included on some modern PCs it was considered nice to produce a USB interface as well and of course the USB can power the circuit. The result of the design effort is that you, the user, may choose between RS232 and USB when it comes to talking to the micro in your Swiss Army Knife.
The Swiss Army Knife board consists of three sections: microcontroller (MCU), RS232 interface and USB interface. Each of these will be discussed separately below.

Microcontroller circuit

The MCU circuit shown in **Figure 1** contains the popular AT89C8252 microcontroller clocked at 22.118 MHz and programmed with a version of Tiny Basic called *Tiny Control Basic* (TCB). The circuit shown is about as minimal as you can get. It is possible to program the code memory allowing a mix of high-level and assembler can be used. To facilitate this IC2 has been included although the chip could be omitted in the finished application or if in-circuit programming is not required.
Regarding connectors, K1 takes all of the I/O lines to a 40-way header and K2 is a 14-way header that mates with either of the two other boards. Pins 1 and 2 form the serial communication. The handshaking pins are not involved in the serial communication.
The AT89C8252 can be programmed serially using just three lines: SCK (clock), MISO (output) and MOSI (input). Using these three lines and sending special bytes it is possible to get direct access to the code memory without removing the chip from the circuit, hence the term *in-circuit programming*. Pin 5 on K1, the DTR line, initiates this process and the other pins are as described in **Table 1**. The DTR signal enables the top half of IC2, a tri-state inverting buffer. All of the lines are inverted by IC2 so the software must take this into account. The ByVac terminal described further on has suitable software built in.
Along with the in-circuit programming there are various other signals available that apply to the USB interface. Pin 6 is the Ring Indicator. Providing the PC is set up correctly and the USB board is also configured correctly, this line can wake up a sleeping PC. By taking the line low the USB chip will send the correct signals to the PC to activate the PC from a suspended state. For this to work properly the microcontroller cannot be self powered, as the current available on standby is insufficient to operate the microcontroller without that also being in power down mode. In the power down mode the microcontroller will not be able to initiate a signal.
The USB specification has strict control over how much power can be taken from it and at what time. A normal USB is capable of providing a limited amount of current (100 mA) until it is properly configured. Once it is, up to 500 mA can be delivered by the USB bus. Pin 7 goes low to indicate that the USB is properly configured. This action enables the lower half of IC2 and as a result the PWR LED illuminates indicating that PWR is active and that current can be drawn from it. Power to external devices is taken from pin 8. The USB circuit (see later) will not provide any power to this pin until it is configured properly.
Pin 9 is a signal from the USB interface that goes low during suspend. The user can monitor this line. Pin 14 is power provided from the USB bus, con-

Features

- Built-in integer BASIC (TCB)
- BASIC can call assembler programs
- Instant code execution, no compiling phase
- Only needs a simple terminal emulator to program
- Auto Baud detection
- User BASIC program, line 10 runs on reset
- Can program code memory in-circuit using ByVac-Terminal
- I/O connectivity
- USB 2.0 option
- RS232 option
- All software available free of charge

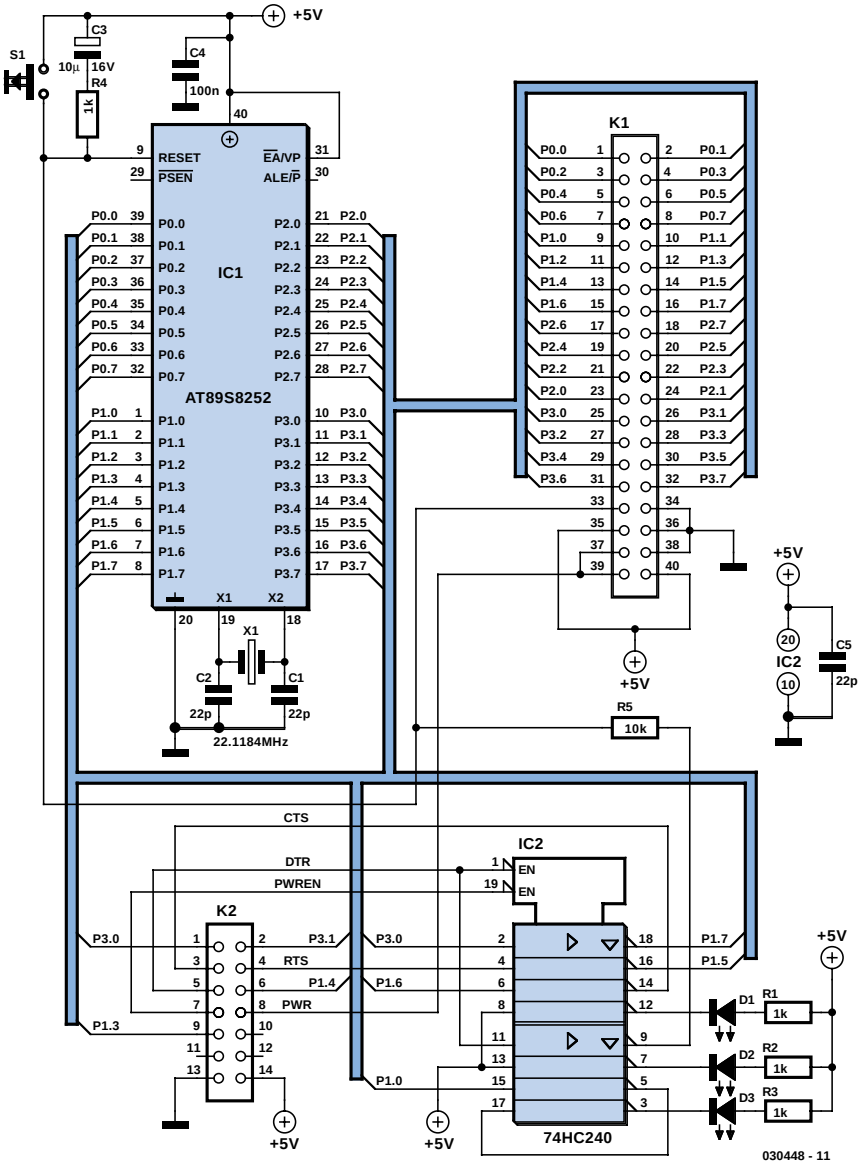


Figure 1.Circuit diagram of the microcontroller main board (MCU). It employs the 89S8252 Flash micro from Atmel.

figured or not. This is only capable of supplying about 100mA when the bus is not configured. For this reason, external devices must use the PWR lines rather than Vcc.

By monitoring the above lines IC2 can be put into power down mode when the PC goes into the suspend state. $\overline{RI}$ Can be used to awaken the PC provided the USB and PC interfaces are set up correctly. To do this IC1 must also be self powered.

The P1.0 LED is connected via two inverters to the P1.0 I/O line of IC1, this can be used for anything. It is useful to have something there for testing purposes.

USB interface

The USB interface shown in **Figure 2** uses a chip from Future Technology Devices Intl. Ltd. (FTDI) and the circuit follows their data sheet. The FT232BM is designed as a quasi-replacement RS232 interface and when installing the device drivers, it becomes a COM port on the PC. Drivers for other operating systems are available. The chip and associated software drivers remove any hassle from building a USB interface.

There are only a few aspects of this circuit worthy of attention. The first is the N channel MOSFET that supplies power to pin 8 of K2 when the $\overline{PWREN}$ signal is low. It is set up this way so that external devices can be powered from the USB bus when the FT232BM is initiated. The initiation process is called enumeration. The other part of the circuit is IC2. This part is optional and is only required if you need to use the special features of the chip, USB 2.0 for example, or are using two chips in the same system. The circuit will work perfectly well without it.

Before going any further it is helpful to know something about the USB interface. The interface is incredibly useful in that not only does it provide a high-speed interface but power as well.

When first plugging in the device it must draw no more than 100 mA until the device is enumerated. Once enumerated the bus is capable of supplying up to 500 mA. In suspend mode this reduces to 500 µA. All USB devices have a unique, registered ID. The FT232BM has a built in, default ID. For

Jim Spence

Jim was born in 1953 and has a degree in Construction (1979). He is currently an IT Project manager working for a global IT company. Jim's last article was published 10 years ago in Electronics Today International. That was another single board computer based on a Z80 which ran Forth. Before managing projects Jim was a Lecturer in Computer Aided Design. His main interests are electronics and software languages. Jim says he goes back to the time of valves and has been lucky enough to watch the introduction of transistors, integrated circuits and then see the personal computer industry grow from nothing.

jim@byvac.com

commercial use the ID must be registered. Also if you intend to use more than one device on the same system then each device must have a unique product ID. It is possible to set all of this up by fitting the EEPROM IC2.

Once enumeration has taken place, which is all taken care of by the FT232BM chip, the $\overline{\text{PWREN}}$ line is taken low. This is passed on to pin 7 of K2 so that it can be monitored by the microcontroller if necessary. It also activates T1 which in turn allows 500 mA to be drawn from the USB bus via pin 8 of K2.

There are two main types of socket: 'A' and 'B'. The 'A' socket is a power provider and is the type fitted to the PC. The 'B' type is a power user and to our knowledge comes in about three different shapes. The one used on the PCB is the most common although smaller ones can be obtained with difficulty, these are usually found on cameras and the like.

The FT232BM requires a device driver, even on an XP system. All the device drivers, software and documentation required for the device can be obtained from the FTDI website. The driver needed is the Virtual COM Port (VCP). This will enable you to use the device as if it were a COM port.

The EEPROM device, if fitted, can be programmed using the D2XX drivers and one of the many software utilities provided at the site. Note **the drivers will no co-exist**, you need to uninstall one to use the other.

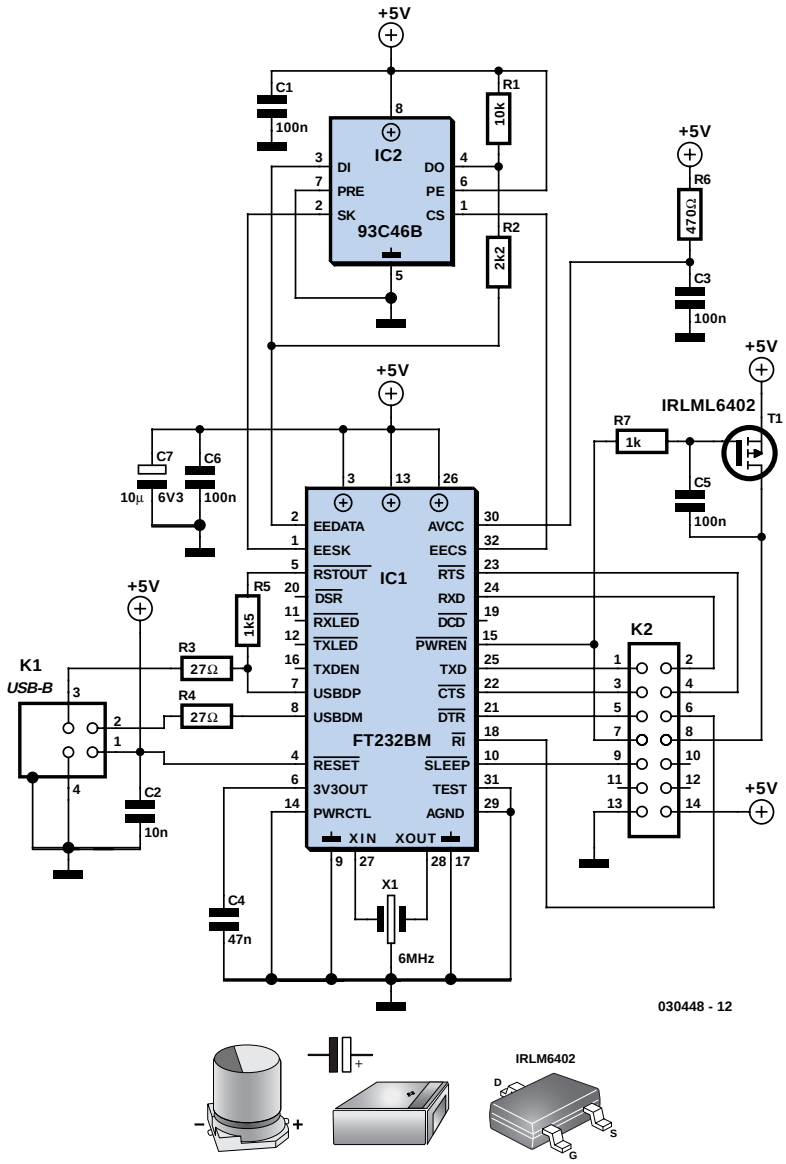


Figure 2. USB interface schematic.

Good old RS232

This is for those of you who do not fancy USB. The RS232 interface circuit diagram appears in **Figure 3.** As RS232 port is not capable of enough supplying power it is necessary to incorporate a simple 5 V supply into the design. Bridge rectifier B1 may strike you as unusual but is well worth the extra cost, not only can you use an AC output power supply but also it does not matter which way round the polarity is on a DC power supply — just plug it in and it works.

Power from the supply is taken to pin 8 on K2 to indicate to the main board that power is available. Pin 7 is permanently at ground indicating to the main board that power is available. The circuit

COMPONENTS LIST

PCB, order code **030448-1**
Project software on two disks, order code **030448-11** or Free Download.

MCU board

Resistors:
R1-R4 = 1kΩ
R5 = 10kΩ

C1,C2 = 22pF
C3 = 10µF 16V radial
C4,C5 = 100nF

Semiconductors:
D1,D2,D3 = LED, low current, colours to personal taste
IC1 = AT89S8252-24PC, Dip40 case, programmed, order code **030448-41**
IC2 = 74HC240
K1 = 40-way boxheader (two pin rows)
K2 = 14-way socket, angled pins, two receptacle rows
S1 = pushbutton, 1 make contact, miniature
X1 = 22.1184MHz quartz crystal

USB interface

All parts SMD, case shape 1206

Resistors:
R1 =10kΩ
R2 = 2kΩ2
R3,R4 = 27Ω
R5 = 1kΩ5
R6 = 470Ω
R7= 1kΩ

Capacitors:
C1,C3,C5,C6 = 100nF
C2 = 10nF
C4 =47nF

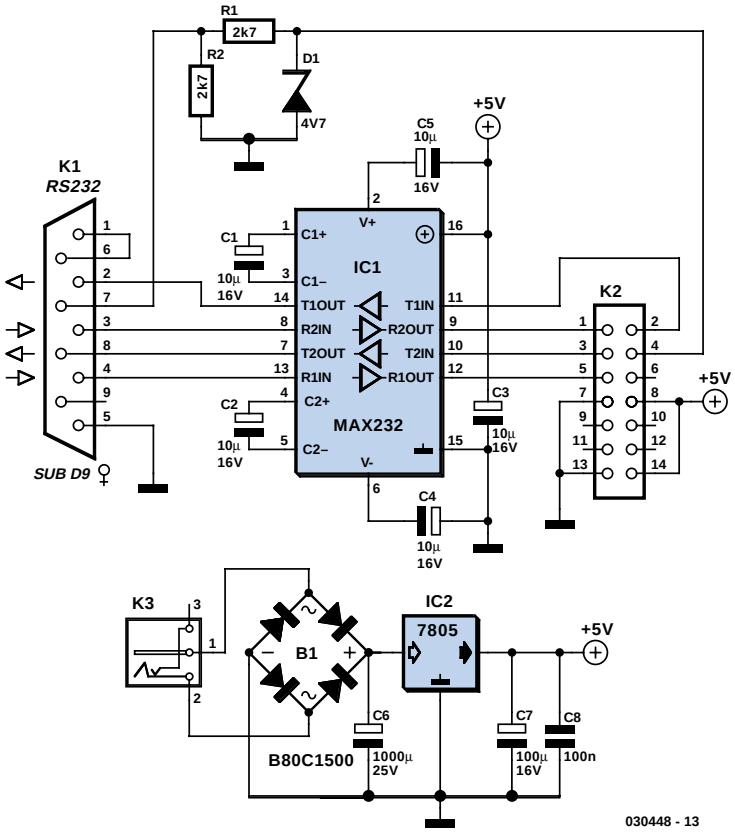


Figure 3. RS232 interface schematic.

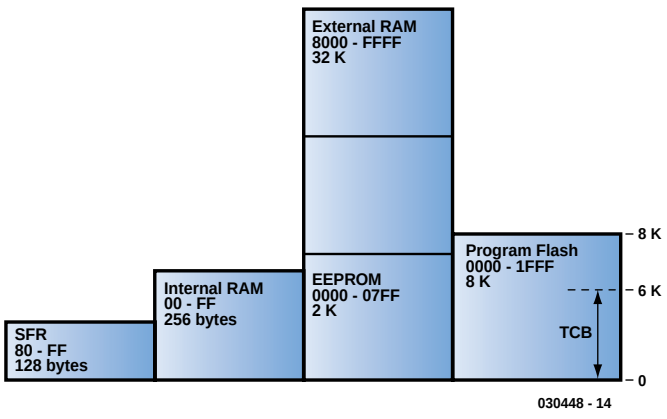


Figure 5. Harvard memory structure applicable to the 89S8252 Flash controller in this project. TCB occupies 6k of the 8k available in the Program Flash area.

formed by R1, R2 and D1 limits the voltage on the RTS line and make it suitable for feeding into IC1 on the main board.
All of the I/O lines are used on IC1 in order to facilitate the in-circuit programming. Only TXD and RXD are required if the in-circuit programming is not required.
Some of the functionality of the USB board is not available to the RS232 interface, so pins 6 and 9 of K2 are left unconnected.

Construction & test: MCU and RS232...

If you look at the PCB artwork shown in **Figure 4**, you'll notice that the MCU, USB and RS232 sections are supplied as one board, order code **030448-1** from our Readers Services.
Depending on the connectivity you have in mind for the Swiss Army Knife you will have to populate the USB or the RS232. Sure, you can build both sections but do remember you can't use them at the same time.
The MCU and RS232 sections are nothing special when it comes to building them — simply work along the lines indicated by the parts list and the component overlay. Sockets are of course recommended for the ICs. The 14-way connector can be cut from a larger one if required. Voltage regulator IC2 does not require a heat-sink.
Before inserting the ICs into their sockets, connect the MCU board to a 5-V supply and power up. Check with a meter that 5 volts appears at the correct polarity across pins 20 and 40 for IC1 and 10 and 20 for IC2. If all is well disconnect the power, insert the IC's and re-connect the power. If you have a logic probe or oscilloscope, monitor pin 2 on K2 (pin 11 on IC1). Press reset and after 1 or 2 seconds the sign-on signal should be seen. This will be a short burst of pulses at 9600 baud.

C7 = 10µF 6.3V SMD

Semiconductors:
T1 = IRLML6402
IC1 = FT232BM (FTDI, www.ftdichip.com)
IC2 = do not fit (93C46B SO8)

Miscellaneous:
K1 = USB connector, type 'B', PCB mount
K2 = 14-way socket, angled pins, two receptacle rows
X1 = 6MHz ceramic resonator, 3 pins

RS232 board

Resistors:
R1,R2 = 2kΩ7

Capacitors:
C1-C5 = 10µF 16V radial
C6 = 1000µF 25V radial
C7 = 100µF 16V radial
C8 = 100nF

Semiconductors:
B1=B80C1500, round case (80V piv, 1.5A)
D1 = 4.7V zener diode, 500mW

IC1 = MAX232 (Dip16 case)
IC2 = 7805

Miscellaneous:
K3 = mains adapter socket, PCB mount, angled pins
K1 = 9-way sub-D socket (female), angled pins, PCB mount
K2 = 14-way angled pinheader, two pin rows

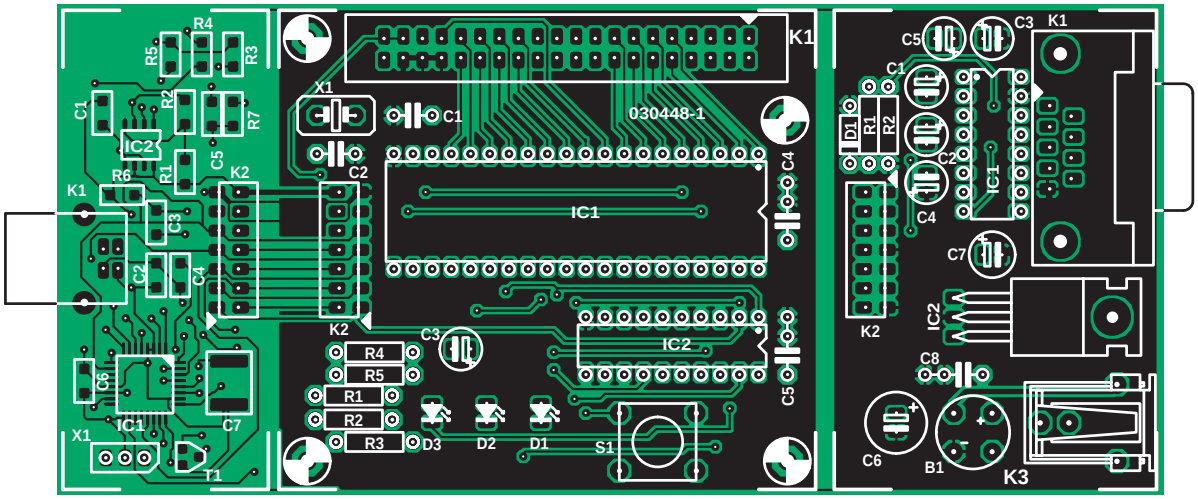


Figure 4. Component overlay for the combined MCU / RS232 / USB board, order code 030448-1. If you want to use USB either leave the USB interface section attached to the MCU or separate the two and connect them with a flatcable.

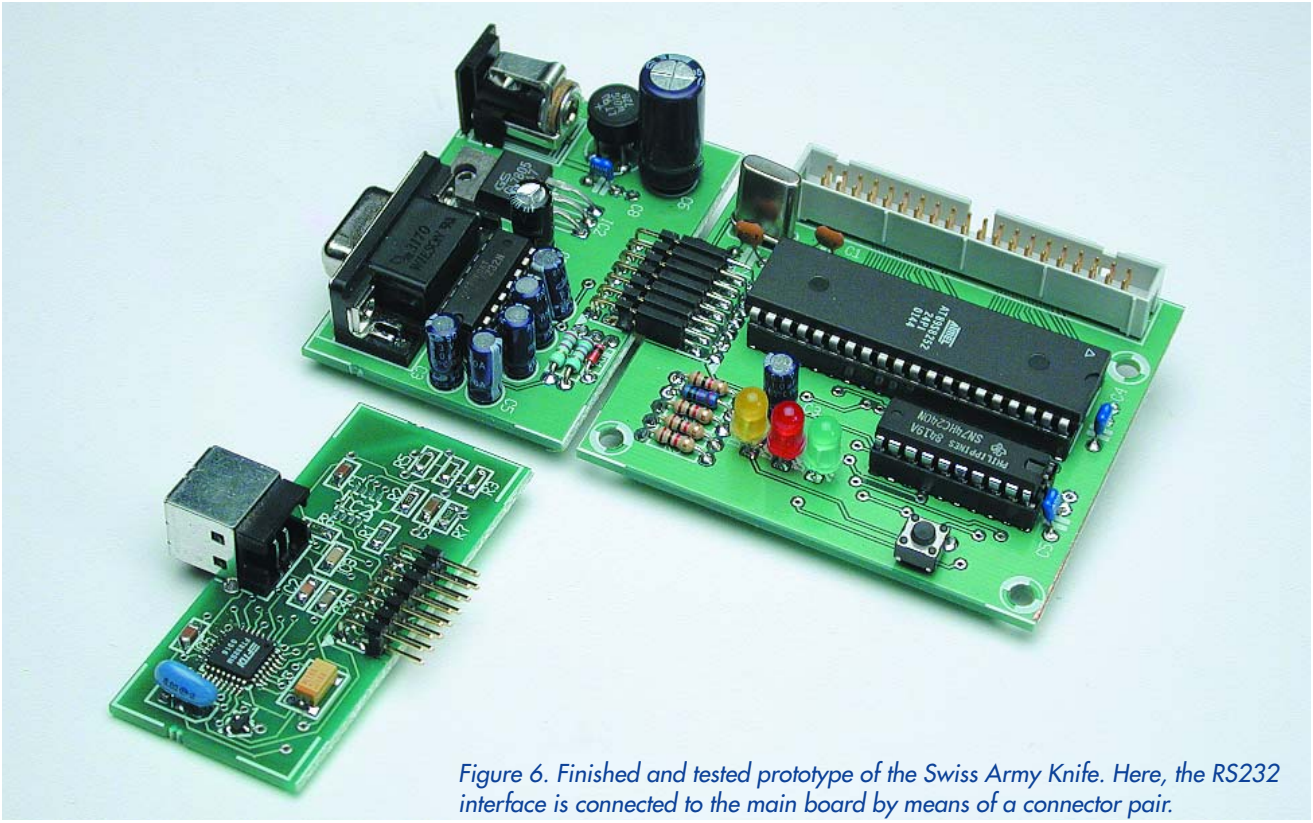


Figure 6. Finished and tested prototype of the Swiss Army Knife. Here, the RS232 interface is connected to the main board by means of a connector pair.

ByVac-Terminal

ByVac-Terminal does have some built-in features to get maximum performance and ease of use from the Swiss Army Knife. Version 1.0 is free from the Free Downloads section of our website: www.elektor-electronics.co.uk. As an example, to download a program from your text editor you can simply use 'Send File'. If however you are using the EEPROM memory space then there is a noticeable delay while the program line is written. An option on some terminal emulators is to send line by line and insert a delay between lines. This works fine but you have to cater for the worst case meaning the download is much slower than it should be.

Built into TCB is the 'LOADB' command that has a very simple protocol and it works like this. After issuing the command TCB waits for a line of BASIC to be sent, when the line is received it is processed and an ASCII code 6 (ACK) is sent back to the terminal to indicate it is ready for the next line. This simple protocol works exceptionally well and the ByVac terminal has this protocol built in.

Another advantage is that it is capable of using the in-circuit programming features so assembler code can be written and downloaded into the code space.

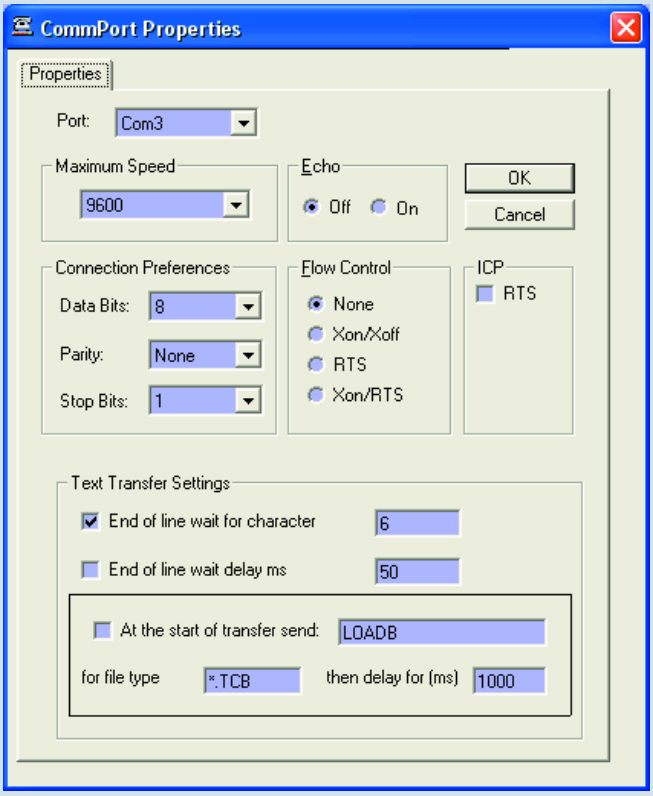
We've already seen how to turn the on board LED on and off using a simple program that you typed in at the terminal. You can verify that everything is okay by typing:

Pz1=254

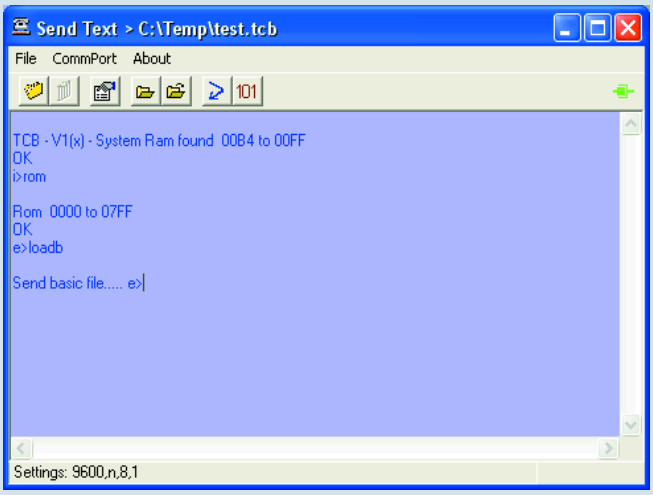
Pz1=255

The above action should have turned the LED on and then off again. You can edit TCB programs by re-typing the line, to remove a line simply type the line number. This however can become very tedious. A much better way is to create and edit the program using "Notepad" or a similar text editor and then downloading this to the board.

Open your text editor and enter the program below. This is very similar to the program you used in the TCB introduction. Note that at lines 20 and 40, instead of assigning a value directly to port 1 a logical operator ('and' 'or') is used. This will have the same effect but it does not affect any of the other lines on the same port. Observe also the



A



B

It goes without saying that IC1 should be pre-programmed with TCB (Tiny Control BASIC) for the above to work.
There is not much else you can do without the other two boards; the MCU circuit is so simple that just about the only thing that can go wrong is the soldering and component location (is everything in its correct place and the right way round).

...and USB

As you can see from the PCB artwork, the USB interface is connected to the MCU section by copper tracks. If you want to fit the USB interface at some distance from the MCU board, the PCB sections have to be separated by cutting or sawing and a small connection cable installed between the respective connectors.
Now for some bad news. The FT232BM chip only comes in a surface mount version and it's a small one at that. You need to be brave to build this but it can be done with a simple soldering iron, solder wick and some solder paste.
We would urge you to have a go at this, it's not a beginner task but it's not impossible either.
Unfortunately, solder paste is expensive but it does make the process much easier. An alternative to using the paste is to use far too much solder and get rid of the excess with the solder wick.
The solder wick does such a good job at tidying up that it is not that important if the solder goes in all the wrong places at first. The most important thing to get right is the orientation of the IC1. It must be perfectly in line and square with all of the pads and remain there whilst the first heat is applied. If it slips then you're in trouble.
We would recommend that you spend a lot of time lining IC1 up and when you are satisfied, solder just a few pins, (one pin if you can) the minimum coverage of the author's fine tipped soldering iron is two to three pins. If all is well solder the rest of the pins, once in place it is almost impossible to remove. Remove all of the excess solder with solder wick and double check no short circuits exist between the pins.

"end" without a line number at the bottom of the program. This will tell TCB that the download has finished. In practice, if you forget to do this it still works okay.

```
10   for j = 1 to 10
20   pz1 = pz1 and 254
30   gosub 500
40   pz1 = pz1 or 1
50   gosub 500
60   next j
70   end

500  for k = 1 to 20
510  next k
530  return
end
```

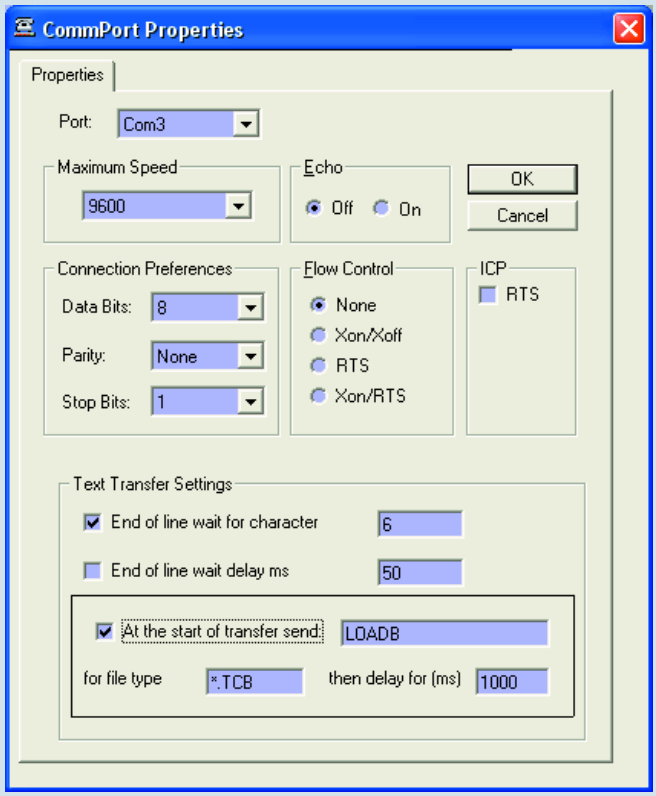
Save the program with the extension TCB, e.g. "LedFlash.tcb" If you are using Notepad be careful to select all file types before saving otherwise you end up with a file called "LedFlash.tcb.txt". This only happens on the first save but it is something to be aware of.

As indicated TCB has a very simple but effective file transfer protocol that the ByVac-Terminal takes advantage of. Start the terminal and use the initial settings as shown in **Figure A**. Note that the "End of line wait for character" is checked. At the terminal type LOADB as **Figure B**.

TCB is now ready to accept a basic program using the simple protocol. Use File and "Transmit text file" or use the forth icon from the left. Select your file "LedFlash.tcb" and it will be loaded into TCB.

The program will load, type RUN to see the fruits of your efforts. To speed things up if you alter the text transfer settings to that of **Figure C**. The terminal will now type in LOADB for you if you select a file with an extension "tcb". Changes to the RS232 settings require the connection between the PC and the Swiss Army Knife to be broken and then restored again.

Use the fifth icon from the left to reload the same file. The



C

cycle for development then becomes:

1. Edit program
2. Save
3. Use the reload icon
4. RUN
5. Back to 1

If you want to stop the program running use CTRL-C or simply press reset. The beauty of an interactive system such as this is that everything is immediate, you can try things out at the command line and it will be instantly activated. At the end of development you have your finished product.

Two dice programs, one simple and the other, well, slightly less simple, to test the above sequence may be found on our website.

The other components are not too bad. They were chosen for their large size (relatively speaking). The connector K2 is a pinheader that will probably need cutting from a larger one.

Before plugging in the device to the PC, download the FTDI device drivers and unzip to a suitable directory. If all is well when you first plug the device in you will be asked for the device driver. Take some time after building to inspect the circuit for shorts, use a meter if necessary. Plug the device into the USB port on the PC or preferably onto a hub to prevent any possible damage to the PC. Although the USB specification calls for short circuit protection, you never know, and a new hub is much cheaper than a PC. If it's any consolation the prototype circuit was not checked well enough the first time it was plugged into a PC and all the USB devices suddenly stopped working, this included the mouse. It took a reset of the PC to restore things back to normal — not a pleasant experience.

Assuming all is well the PC will detect the new device and request the location of the device drivers. Install this just as you would any device driver. If you are not sure how to do this look on the FTDI website for information, installation instructions also come with the device drivers for Windows and other operating systems.

All being well you should now have a new COM port. To find out which port number has been allocated depends on the operating system. For Windows XP this is in the Control Panel → System → Hardware → Device Manager then open up the Ports tree by clicking on the + sign. You should see a new port. If not, re-install the device driver and make a note of any error messages.

Complete circuit test

Connect the two boards together, use either the RS232 circuit or the USB circuit. Launch the free ByVac terminal utility, see the 'ByVac-Terminal' inset. If using the RS232 board you need a straight-through (1:1) cable where pin 2 goes to pin 2 and pin 3 goes to

Table 1 Programming & USB interface

J1 Pin	Name	Function
1	RXD (SCK)	Part of IC1 serial, in-circuit programming interface. This is the clock line.
2	TXD	Serial output from IC1
3	CTS (MISO)	Output from IC1 serial programming interface.
4	RTS (MOSI)	Input to IC1 serial programming interface
5	DTR	Low activates programming mode, also low and back to high again will reset IC1
6	RI	Take low to wake PC (requires special set up)
7	PWREN	Low o/p from USB interface indicates that 500mA is available on the PWR line
8	PWR	5V @ 500mA
9	SLEEP	Goes low to indicate that the connected PC has gone to sleep
10-12		Not used
13	GND	Ground
14	VCC	100 mA, 5V

pin 3 etc. In some cables the pins are crossed, check this with a meter. You also need to enable RTS in the ICP box. Press reset (S1), wait for a few seconds and you should see the sign-on message. If not, check that the sign on message is coming from IC1. If there is a signal and you still do not see the sign-on message check the settings, Baud rate etc., check the connectors, cable and wiring.

Here's TCB

At the heart of this project is the TCB (Tiny Control BASIC) software that will incidentally run on any 89C8252 system with or without external RAM. If there is any RAM present it will automatically detect it, obviously there is no RAM in this project.
There are three ways you can get the TCB software onto IC1:

1. use the programming interface;
2. buy a pre-programmed chip, order code 030448-41 from Readers Services;
3. use an 89C8252 programmer.

Although possible, option 1 is not recommended, the programming interface and software being designed for short programs(<100 lines), also it will mean that you can't properly test the circuits until you have done this so how do you know if the circuit has been built correctly? It is possible however but will take about 25 minutes, see the Assembly Code inset. Options 2 or 3 are recommended if you are building the circuit for the first time.
This processor and its architecture have been mentioned many times in various articles, however the memory aspect must be at least partly understood so it is briefly discussed here.

Memory

The memory space of the 8052 uses an architecture that shares parallel areas of memory. By logically combining the OE and PSEN signals to access the RAM it means that the external RAM can be used as program memory. The 8051 architecture is old and we're sure that separating out the data and program memory seemed like a good idea at the time. This is what is known as Harvard Architecture and in theory at least it means that it is possible to fetch a code instruction at the same time as data in a single machine cycle.

Figure 5 shows the memory map and may cause some confusion for those of you who are more used to the conventional memory arrangements. The EEPROM space is pure data memory and assembler programs cannot run in this space, high level programs can, see later. There is provided 256 bytes of internal RAM that shares the upper half of this space with special function registers. The trick to 'getting at' the various spaces lays in the instruction set. It would take up too much room here to go into detail but to give an example when accessing the code memory the MOVC instruction is used but when accessing external memory MOVX is used. Other techniques of direct and indirect addressing are used to access the internal RAM spaces.
The on-board Flash memory contains TCB, version 1 and only occupies about the first 6 k. If this is good enough for your applications then you can forget all about the various memory addressing modes, TCB will take care of it.

Introduction to programming in TCB

The integer BASIC is a modified version of Tiny BASIC called Tiny Control Basic (TCB) that was designed specifically to enable users to get the maximum out of a microcontroller in the shortest possible time without having to go through a massive learning curve, or installing any special software.

The 8052 architecture has three memory spaces that are dealt with by TCB. Internal RAM, EEPROM and external RAM. These are accessed by the keywords IRAM, ROM & RAM respectively. Type IRAM and you should see this message:

```
i>iram

Internal Ram  00B4 to 00FF
OK
i>
```

The 'i>' prompt indicates that we are now in internal ram space. As you can see there are only a few bytes but this is good enough for a 2 or 3 line program. TCB will let you know if you run out of space. To access the EEPROM type ROM:

Table 2. Tiny Control BASIC main specification	
Numbers:	16 bit signed integers range from –32767 to 32767
Variables:	single letter A through L (12)
Arithmetic:	+, -, *, /, and MOD
Logic:	NOT, AND, OR, XOR
Comparisons:	>, <, =, <>, >=, <=
Commands:	RUN, LIST, NEW, DUMP, RND, ABS, IF, THEN, GOTO, FOR, TO, NEXT, REM, CALL, RETURN, GOSUB, ROM, RAM, IRAM, LOADH, LOADB, DECIMAL, HEX, LET, PRINT, INPUT, BAUD
Special function registers:	PZ0, PZ1, PZ2, PZ3, TCON, TMOD, TL0, TL1, TH0, TH1, SCON, SBU, IE, IP, T2CON, WMCON, SPCR, SPSR, SPDR, PCON
Interrupt:	ONEX0, ONEX1, ONT0, ONT1, ONT2, ONSP, EI, DI

```
i>rom

Rom  0000 to 07FF
OK
e>
```

We are now in ROM space and as you can see there is 2 k of memory from 0000 to 07FF. At any time you can type 'DUMP' to see the contents of the memory. The advantage of using this space is that it will retain the program even after power down. The disadvantage is that it is slower than RAM to write to and there is a limit to how many times you can write to it, approximately 100,000 times. In this project there is no external RAM so typing RAM will return an error. All TCB programs will be written to the EEPROM. TCB is capable of running a program at start up but it must begin with line 10. If line 10 does not exist any the program will effectively be erased. The start up procedure is as follows:

1. Check for input from user (space bar) – waits about 1 to 2 seconds
2. If no input, check ROM space for a program starting at line 10
3. Run it if it exists, if not check RAM space for a program starting at line 10 and run that.

If no line 10 exists and there is RAM it will erase (NEW command) the program in RAM and the sign on will be in RAM space. If there is no RAM then it will come up in internal RAM space (IRAM). Memory will not be erased if you press the space bar within one to two seconds of switch on. In this project of course there is no external RAM so the default will be to come up in internal RAM space "i>" unless there is a line 10 in ROM space in which case it will run the program there.
If the space bar is pressed within approximately two seconds of reset or switch on, TCB will detect the Baud rate and come up in internal RAM space.
TBC does not actually erase all of the RAM but simply puts the end of program marker (FF) into the first byte. You can verify this using the DUMP command after the NEW command.
Referring to the MCU circuit, LED D3 is connected to port line P1.0 via two buffers in IC2. By default it will be off. This is because at start up all of the port lines are taken high.
To turn on the LED simply type: PZ1=254

```
i>pz1=254
OK
i>
```

PZ1 is the port 1 variable, anything you set this to will occur on port 1. By setting port 1 to 254 which is 1111 1110 in binary it will set pin 0 of port 1 to 0. By convention this pin is referred to as p1.0

Try:

```
ROM
10  FOR J = 1 TO 10
20  PZ1=254
30  GOSUB 200
40  PZ1=255
50  GOSUB 200
60  NEXT J
70  END

200 FOR K = 1 TO 50
210 NEXT K
220 RETURN
```

The above should flash the LED on and off 20 times.
Table 2 gives a brief description of the language.

Odds & ends

The massive amount of documentation Jim produced for this project would easily fill half this magazine, hence some items had to be moved to our website from where they can be downloaded free of charge. The items include the illustrated *Quick Start Guide — Swiss Army Knife*, the *Tiny Control BASIC Manual* and *Simple Dice*, so get downloading...

(030448-1)

Web pointer

FT232BM USB drivers: www.ftdichip.com

Free Downloads

Byvac-Terminal for PCs (install file with supporting OCX files), TCB (hex file), File number: 030448-11.zip

Two simple dice programs (Word file). File number: 030448-12.zip

Quick Start Guide for Swiss Army Knife (Word file).
File number 030448-13.zip

Tiny Control BASIC manual (pdf file).
File number: 030448-14.zip

PCB layout in PDF format.
File number: 030448-1.zip

www.elektor-electronics.co.uk/dl/dl.htm, select month of publication.

Assembly code

At some point in time more control or faster speed may be required than a high level language can give. There is no alternative but to resort to use assembler, which is the direct equivalent to programming the processor itself.

To do this effectively you will need an assembler. This is a program that translates 2 to 4 letter mnemonics into numbers that the processor can understand. There are many available free for this processor. In the examples shown "ASM51" has been used. It can be downloaded from various sources.

RAM would be required to run assembly code using the LOADH feature but none is available in this project. However, we have free code space in EEPROM starting at 1700h which is accessible through the in-circuit programming feature.

Getting Started

Always the hardest part? You can't really program in assembler without knowing something about the processor and for this you will need at least a data sheet for the processor and some knowledge of the 8051 / 8052 instruction set but the following example will get you on your way.

Example program

We will keep on familiar ground by flashing the on board LED again. As you know, to do this we need to set P1.0 to zero and back to 1 again.

```
; Example of flashing an LED
; Use Asm51
; P1.0 has the LED connected to it
;
$MOD8252

        cseg
        org     1700h
flash:  ; pulse P1.0 up and down
        setb    p1.0
        call    delay
        clr     p1.0
        call    delay
        jmp     flash
;
delay:  ; delay
        mov     r0,#0f0h
del2:   mov     r1,#0
del1:   nop
        djnz    r1,del1
        djnz    r0,del2
        ret

        end
```

The operation of the program is conveniently explained by means of a table.

`&MOD8252`	First of all this contains all of the special names associated with that processor so this needs to be in place for our 89S8252.
`Cseg`	Tells the assembler that the following is code, there is a corresponding "dseg" for data.
`Org`	This is the program origin address. TCB finishes at about 1650h so we don't want to be any lower than this otherwise we will damage TCB.
`setb p1.0`	Sets the bit 0 of port 1 to high, just as pz1=1 would do.
`Call`	Is the equivalent of gosub
`delay:`	This is a subroutine, notice the ':', asm51 requires this. Because of the raw speed of the processor 2 delays are required. In fact this carries out over 61,000 instructions before returning which gives a delay of about 0.25 second.
`R0,#0f0h`	R0 is a general purpose register of which there are 8, R0-R7. There are in fact 4 banks of them but only one bank can be used at a time. What this instruction does is to place the value of F0 (hex) into register R0. The '#' is known as an 'immediate' modifier. If you did R0,0f0h then the contents of memory address F0 would go into R0. (yes there is plenty of scope for error).
`Djnz`	This is Decrement and Jump if not Zero so on the inner loop R1 will be decremented and if it is not zero then the program will jump to the label "del1:". Something to note here is that the register is decremented before zero is tested so a register starting out with a value of zero will be decremented to FF before being tested.

Assembling

Use Notepad or any text editor to create the above program, save it with an 'asm' extension, for example FlashLed.asm. Now that the program is written it needs converting to a format that the processor can understand. To do this you will need to use the command prompt if you are using the Windows operating system. If you are using DOS then you are already there.

Choose a directory to put the files in, you should have FlashLed.asm, Asm51 and MOD8252 in the same directory (unless you know how to set paths up). At the prompt type "asm51 FlashLed". On pressing enter you should see something similar to that of **Figure A**. Note that Asm51or indeed any other 8051 assembler has to be obtained separately, it is not included with the software for this project.

This process produces two files using the same file name but with different extensions thus: FlasLed.hex and FlasLed.lst. The LST file is where you will find any errors if there are any and the HEX file is a special format developed by Intel to enable code to be loaded into processors. As a matter of interest the format consists of lines of text preceded by a colon.

```
:10170000D29012170CC29012170C80F478F0790066
```

The first two numbers '10' is the length of the code in hex, the next 4 numbers is the memory address, in our case 1700 this is followed by a record type which is almost always 00. The rest of the line consisted of the actual code to be loaded at the given memory address except for the last two numbers, '66' which is a checksum to verify that the line has been received correctly.

Load Up

The next job is to load this program into code space. This is where the special features come in. Click on the "101" icon and the background will turn black. We are now in flash programming mode. Press 'p' and the text will turn yellow, this indicates that we will be using Program or Code space (the flash memory area). Typing "v" "88 <enter>" and then "99<enter>" will enable you to view the contents of the code memory from memory address 0088 to 0099. This is where the sign-on message of TCB is located. All numbers are in hexadecimal format, see **Figure B**.

To program this area with our assemble program type "h" This will pop a dialog box that will allow you to select the FlashLed.hex file. This will be loaded into the memory and you should get something like **Figure C**. Programming will now commence automatically (and it does take some time to complete).

Running the program

First of all get back into terminal mode by pressing 'x' or the '101' icon. To run this program TCB has provided CALL. The syntax is CALL n where n is the address you want to jump to. The address we chose to put the program at was 1700h, TCB works by default in decimal, to change this type HEX. Another word of caution, everything is in hex from now on until you reset or type DECIMAL. This applies to normal basic program, hex line numbers look very strange.

Type the following to run the program:

```
HEX
CALL 1700
```

If you don't want to use HEX then CALL 5888 works just as well, 5888 is the decimal equivalent of 1700h.

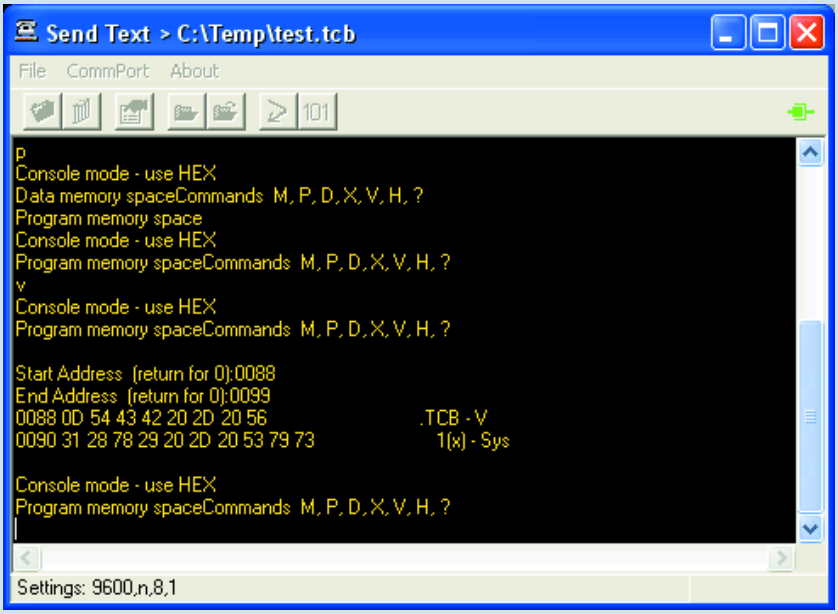
The LED should now flash indicating that you are running the assembler program. Note because this is such a simple program, the only way to stop it is to press reset. Do this at the terminal using the icon next to "101" or using the push button on the main board. Once in flash memory, the program will remain after power is removed. Because of the slow serial programming nature it is not really suitable for large files. Although this can be done, expect TCB to take about 25 minutes for the basic.hex file. Some things simply can't be done using TCB alone, this feature makes the project completely versatile.

You can quite happily use a mixture of TCB and assembler. You can for example assemble the code at address 1700 and use a simple 2 line basic program:

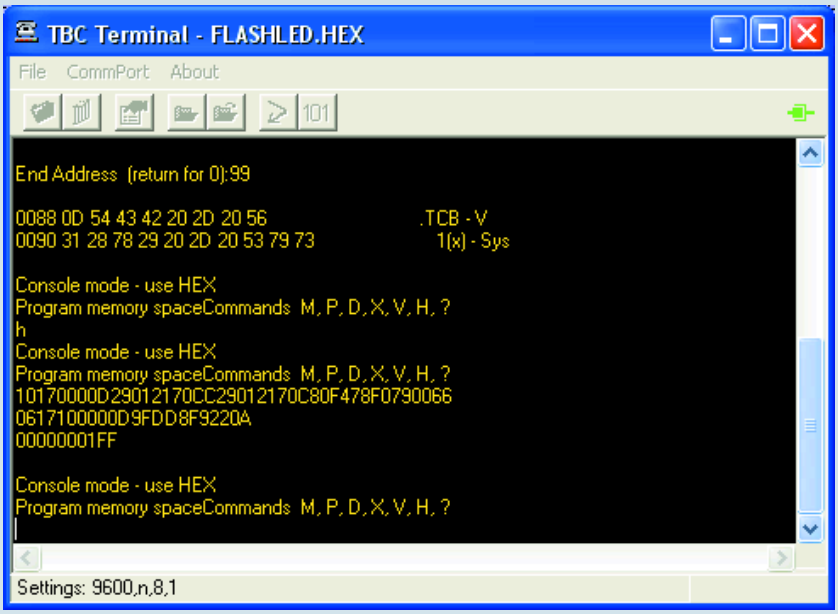
```
10 HEX
20 CALL 1700
```

This will start flashing the LED whenever there is power applied to the board. Okay so a flashing LED has its limitations but we're sure you get the idea.

A



B

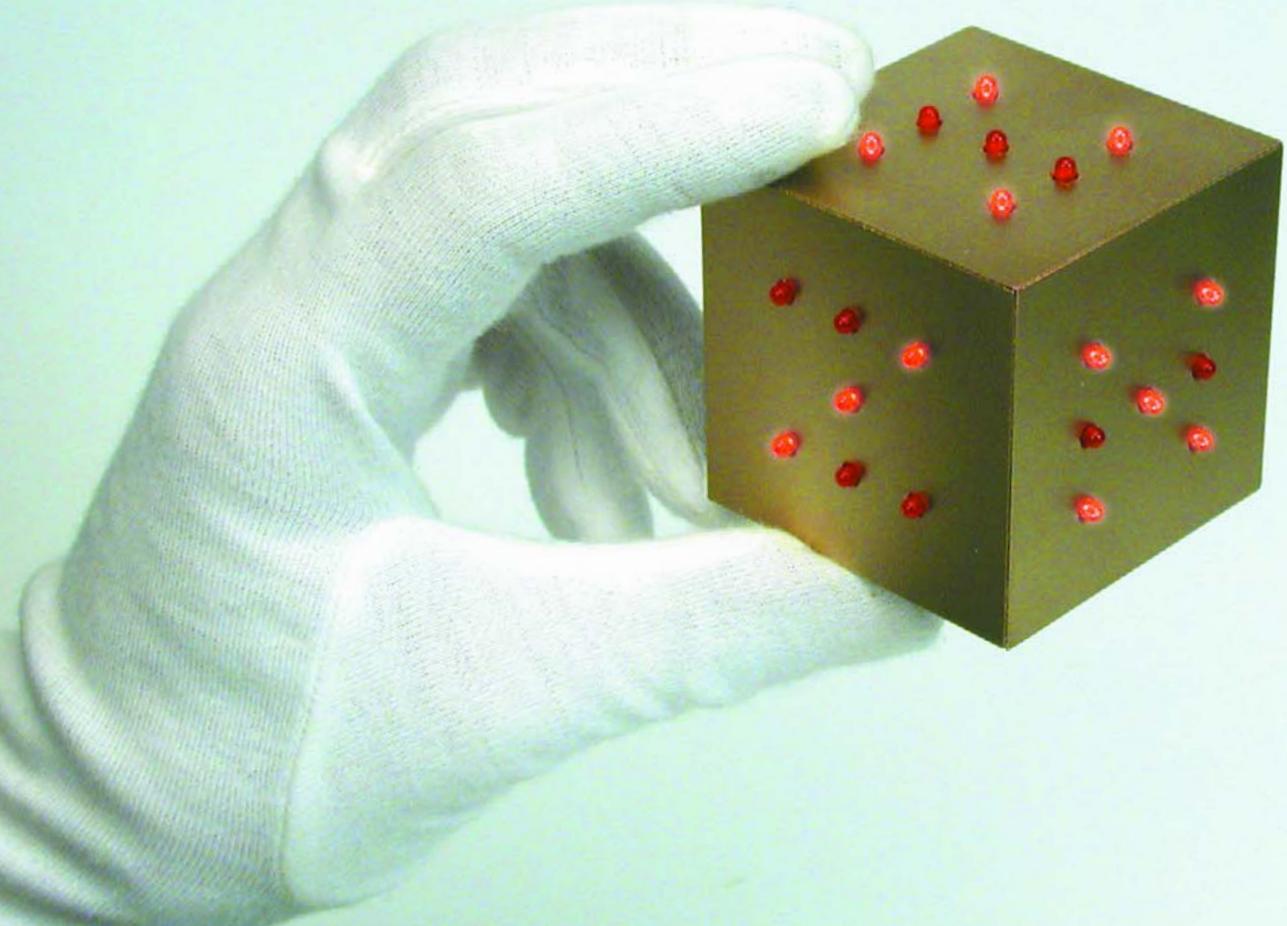


C

Rolling

Paul Goossens

Projects for electronic dice have previously appeared in Elektor Electronics. Different designs and shapes have found their way into the magazine, but one thing they had in common: they were all two-dimensional dice. But now that is about to change!



Dice

Three-dimensional electronics

Electronic dice are usually constructed on flat boards. To give the impression of a 'real' dice, seven LEDs are often used in a pattern similar to that on a dice. The original intention was to base this circuit on a traditional 'flat' design, until somebody from the editorial staff suggested that you should really be able to throw a dice. After all, a dice is not an item that lies still on the table.

'But that is the advantage of an electronic dice!' you'll say. You don't need extra space to throw it and it won't roll off the table either. And this was exactly how our design staff reacted to this unusual suggestion. But the editor was adamant: this dice has to roll!

Fortunately the team at *Elektor Electronics* handles such situations smoothly and it wasn't long before the lab were prepared to have another look at it.

It didn't take long before the first ideas were put on paper. One obvious design for an electronic rolling dice is to mount LEDs onto a cube, showing the proper value on every side. The LEDs are driven via current limiting resistors from a 9V battery and Bob's your uncle! It quickly became apparent that with a homemade dice it was extremely difficult to get the centre of gravity exactly in the centre. Such a dice would therefore have a bias. The chance that the lightest side ends on top is greater than for any of the other sides. The value on that side will therefore occur more often than the value on the 'heavier' side.

How else?

After some more thought we came to the conclusion that a dice with seven LEDs on every side was the best option. After throwing the dice, the six sides would each take on a different random value. In this way you can guarantee that the dice is completely 'honest'.
This solution does require a bit more electronics. First of all, you have to detect when the dice is in the process of being thrown. For this we used a mercury tilt-switch. This also makes it possible to 'roll' the dice just by shaking it.
The electronics then has to generate the random numbers that appear on the six sides of the dice. The use of a microcontroller comes to mind straight away. This keeps the circuit relatively small, which comes in handy for a dice.
Another part of the design that required some thought is the method used to connect the boards together. It doesn't look very nice if dozens of connections go from a main PCB to the other five boards. To keep the number of connections down we've chosen serial connections between the boards. On every board a shift register takes care of driving the LEDs. This way only four connections are required: two lines for the supply, one for data and the fourth for a clock signal.
Another area that needs to be considered is the current consumption and on/off switch. In this design a standard switch can't be used of course, because nothing may protrude from the dice. It would otherwise roll very strangely and at worst switch itself off. We got round this problem with the addition of a (recessed) push-button to turn the circuit on. It is turned off completely automatically.

Main PCB

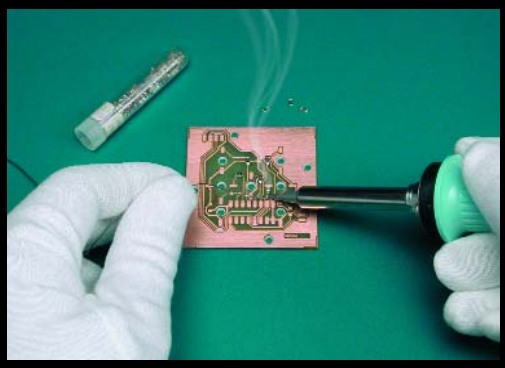
After these considerations we arrived at the circuits for the dice. At the hart of the circuit is IC1, a microcontroller by Atmel (**Figure 1**). We have used this type previously and have given a detailed description in past issues of Elektor Electronics. It therefore suffices just to mention that the software for this project can be freely downloaded from the Elektor Electronics website. As this controller has an internal flash memory for the program there is no need to add external memory. Crystal X1 in conjunction with C2 and C1 provide a clock signal. IC1 drives the 7 LEDs directly via its I/O pins. As we have used low-current LEDs there is no need for an extra buffer.

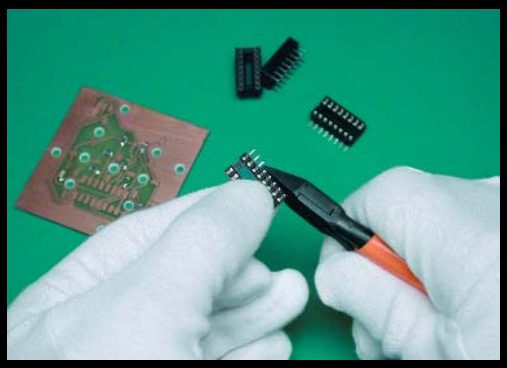
The supply section may appear a little unusual at first. This is because we want to turn the circuit on with a push-button and let the microcontroller itself turn the supply off. The part of the circuit round T1 and T2 provides this functionality. K2 is the connector for the 9 V battery. When push-button S1 on board 2 is pressed, a small current will flow from + 9 V via the base/emitter junction of T1 and R8 to ground. This causes T1 to conduct, feeding a current to voltage regulator IC2. This then supplies the rest of the circuit with 5 V.

When the push-button is released, the current can no longer flow through R8. It is of course undesirable that the circuit would then lose its power. Just try to keep the push-button held down during a throw; it's not exactly user-friendly for a dice.

To get round this problem we've added resistor R9 and transistor T2. T2 is driven via the microcontroller. When the circuit is switched on, T2 is also turned on, causing a current to flow from the base of T1 to ground via R9 and T2. The supply is then no longer dependent on S1 being pressed.
When the processor notices that it hasn't moved for a while (several min-

Figure 3. Construction of the dice.





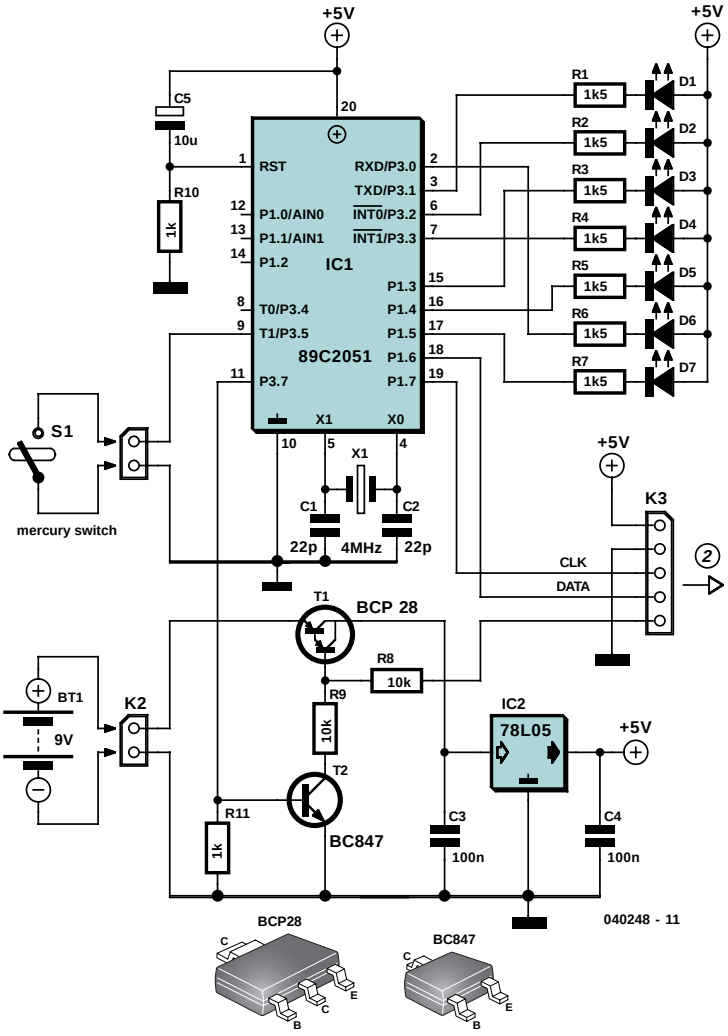


Figure 1. The main board is more complex.

utes), it stops driving T2. The dice then turns itself off automatically. When switch S1 is pressed again, the whole sequence repeats itself.

When the supply is applied a reliable reset signal is generated for the microcontroller by C5 and R10.

The CLK and DATA signals are fed to the next board via connector K3. K3 also has connections for the supply and the on-switch (pin 5). The switch had to placed on board 2 due to a lack of room on the main board.

Other boards

The circuits for the other boards (2 to 6) are almost identical. The main difference is that circuit 2 has a switch, which isn't present on the other boards.

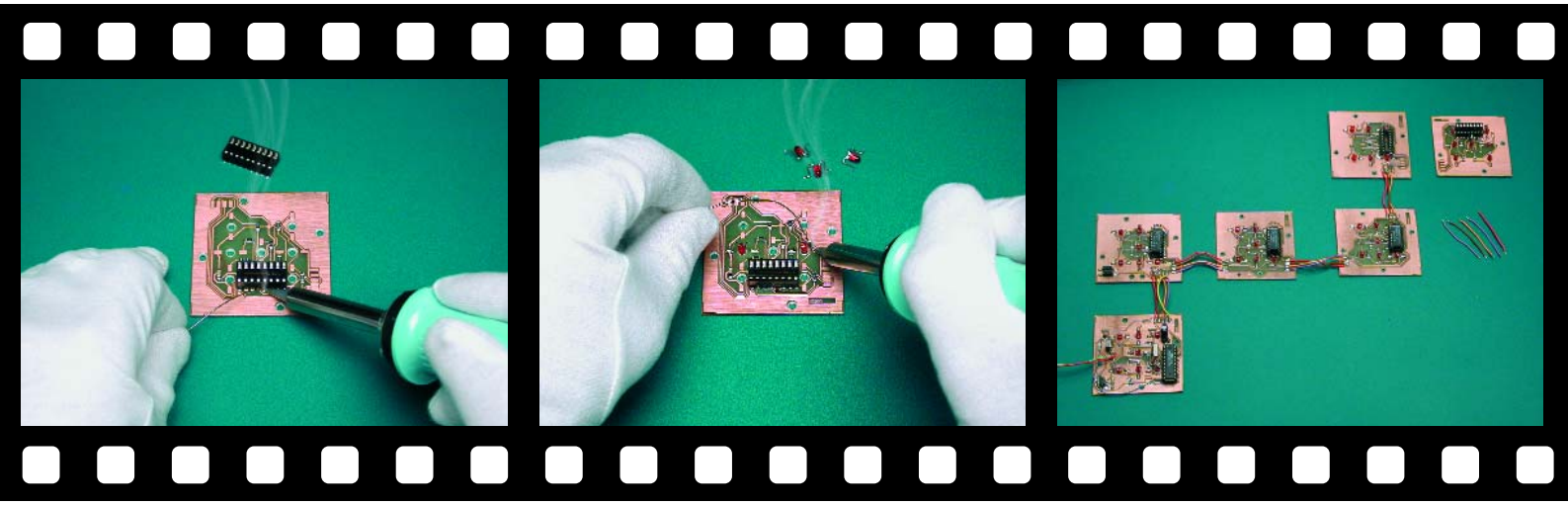
Figure 2 shows the circuit diagrams for board 2 and the other four boards. There is obviously no need for a feed-through connector (K2) on the last board.

At the centre of the circuit is the shift register, a 74HCT4094. This type of IC has a parallel output register. Every rising edge at the clock input causes the 8 bit data in the IC to shift by one bit. The first bit then takes the value of the data input. The last bit shifted out is connected via pin 9 to the data input of the shift register on the following board. This effectively turns all boards into one large shift register.

Construction

The construction of the boards differs in a few ways from the usual method. In several places we have used standard components, which have to be surface mounted. The reason for this is that we'd rather not have any leads sticking out through the boards. This would result in a number of sharp points on the outside of the dice, which would damage your furniture when the dice is thrown, and this obviously isn't our intention. To make things clear, we've included a series of photos of the construction of the prototype in **Figure 3**, as a picture is often worth a thousand words.

The component layouts for all boards are shown in **Figure 4**. First of all, the SMD components are soldered to the boards. Then it is the turn of the IC sockets, which also need to be surface mounted. You need to bend the pins outwards and cut them to length (take a measurement from the board first!). Next it's the turn of the 42 (!) LEDs. The



leads of the LEDs need to bent in the right form, as shown in **Figure 5**. Take great care that the anode and cathode are positioned correctly on the board. The case of the LED is then stuck through the hole. The ends of the leads should now be flat on the board, where they can be soldered. At this stage it is not important if the LEDs are not all at the same height. After soldering all LEDs they are pushed outwards such that they stick out just a little above the surface of the dice. If we now press the board onto the table, the LEDs should all be at the right height.

Apart from the ICs in DIL packages, there are a few other components that have 'normal' leads. As we don't want the outside of the dice blighted by protruding leads and solder, these also have to be surface mounted. This includes the crystal and the mercury tilt-switch. The leads of these components have to be cut fairly short, and the ends turned through 90 degrees. These ends are then soldered onto the board. Take care that you don't shorten the leads too much!

Connecting the boards

Once all boards have been populated it's time to connect them together. **Figure 4** shows how this is done. You should use individual pieces of wire for this, with a length of about 5 cm. If the wires are too short it makes it more difficult to fix the boards together at a later stage. There would also be a greater chance that one or more of the wires became loose!

Lastly the ICs are plugged into their sockets. (Make sure that you plug them in the right way round!)
Now that the electrical construction is finished, the circuit should be tested. It is wise to do this *before* the dice has been fully assembled. At this stage the electronics are still easily accessible, should you need to make any repairs. After you have carefully checked for short circuits you can connect the 9 V battery. When you press the switch (on board 2), the circuit will come alive and a random number of LEDs lights up on the boards.

If you then shake board 1, the dice will simulate a roll. The numbers on all sides will change regularly. After a few seconds the changes slow down and the result of the roll is shown on the boards.

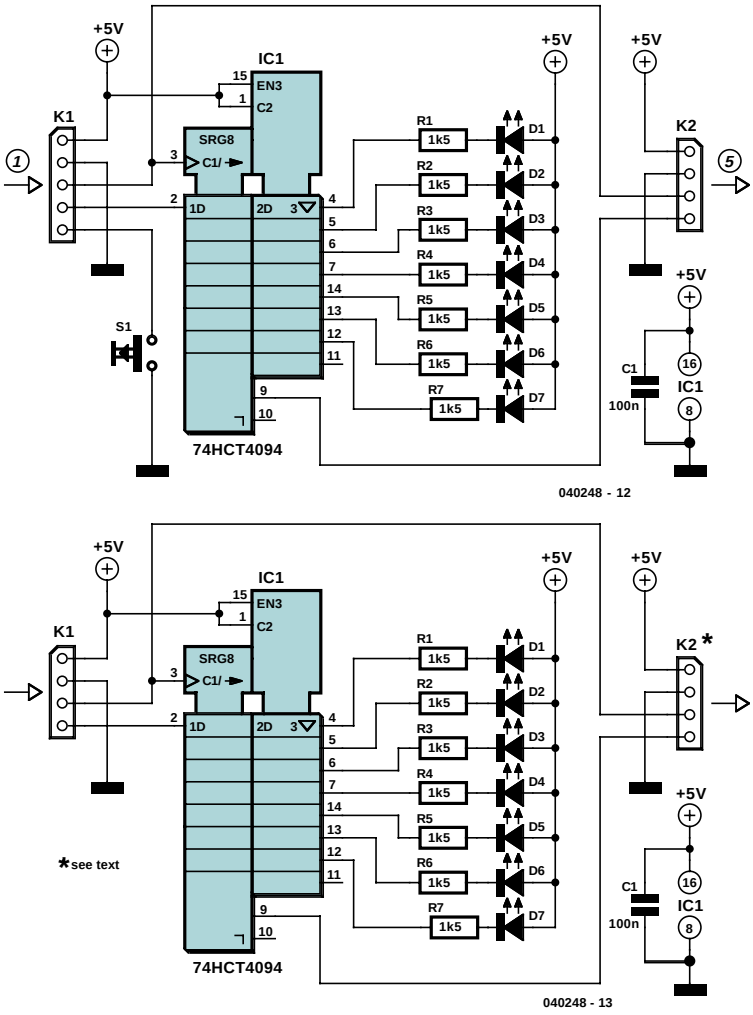


Figure 2. Above is the circuit for board 2, beneath it the circuit for the other four.

3D

The electronics may now be completed, but it doesn't look anything like a dice yet. The boards now have to be fixed together into a cube. Board number 4 is placed flat on the table and boards 2, 3, 5 and 6 are fixed at right angles to it (see **Figure 6**).

Use a plastic angle section (available from most DIY stores) to fix the boards together. You should cut the angle section into small lengths and use these to glue the boards together.

You shouldn't glue the last board (board 1), since this has to be removed every time the battery needs replacing. You need to glue a piece of angle section on the other boards, such that they line up with the 3 mm holes in board 1. Now carefully drill a 2.5 mm hole through each angle section at the point where it meets the mounting hole of board 1, then make a 3 mm thread in it. If you want a stronger thread you can glue a flat piece of plastic onto each angle section, which should be drilled and threaded as well. The inside of the dice should be filled with cotton wool or foam rubber, preventing the 9 V battery from moving or causing a short circuit. You can now screw board 1 into place.

And finally

You could give the dice a nice lick of paint, as we did with our prototype, but we leave that to your own preference. In any case, we hope that you enjoy constructing it and have fun using the dice in many games!

(040248-1)

COMPONENTS LIST

PCB # 2

Resistors:
R1-R7 = 1kΩ5 SMD
R8,R9 = 10kΩ SMD
R10 = 1kΩ SMD
R11 = 100kΩ SMD

Capacitors:
C1,C2 = 22pF
C3,C4 = 100nF SMD
C5 = 10µF 16V radial

Semiconductors:
D1-D7 = LED, 3mm, red, low-current
IC1 = AT89C2051-12PI, programmed, order code **040248-41**
IC2 = 78L05
T1 = BCP28 (Conrad Electronics # 153225-8B)
T2 = BC547B

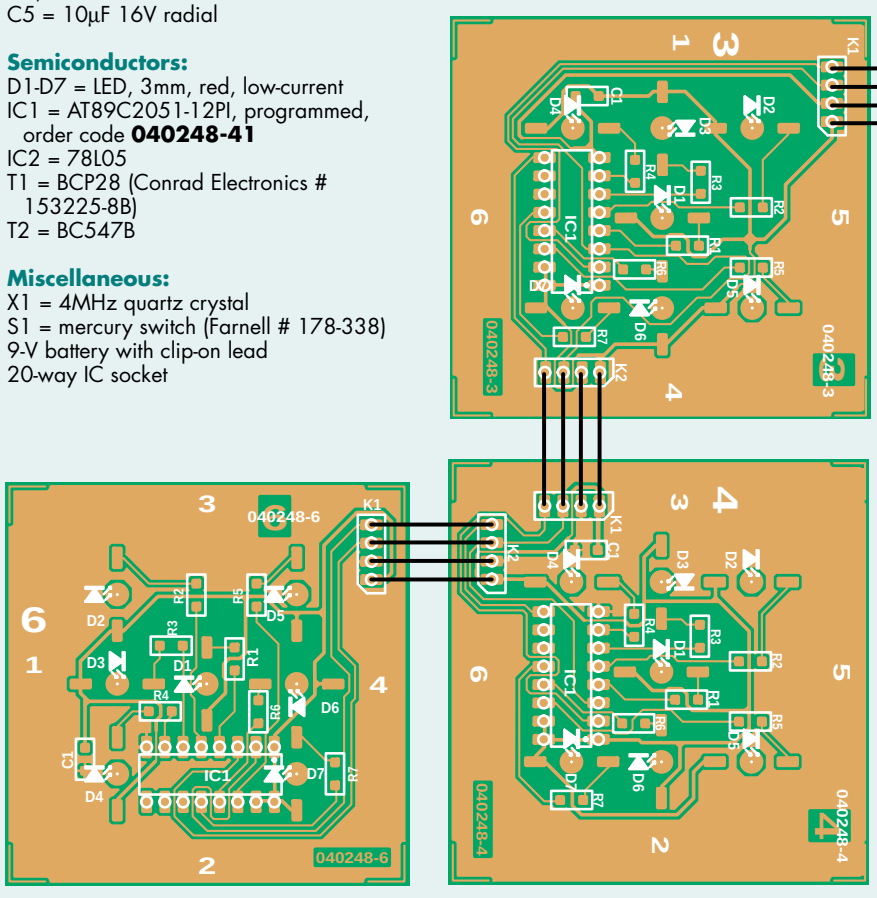
Miscellaneous:
X1 = 4MHz quartz crystal
S1 = mercury switch (Farnell # 178-338)
9-V battery with clip-on lead
20-way IC socket
ABS (hard plastic) angle section
2 M3 screws (countersunk head)

PCB # 2 through # 6
(per board)

Resistors:
R1-R7 = 1kΩ5 SMD

Capacitors:
C1 = 100nF SMD

Semiconductors:
D1-D7 = LED, 3mm, red, low-current
IC1 = 74HCT4094



Order of construction

- Solder the SMD resistors and capacitors
- Bend and cut the pins of the IC sockets
- Solder the IC sockets
- Bend and solder the LEDs
- Prepare the connections between the boards
- Testing
- Connect the boards together
- Add cotton wool and the battery
- Fix the last board into place
- Final test

Miscellaneous:
16-way IC socket
S1 = switch type DTS61K (only on board # 2)

PCB, order code 040248-1 (contains six sections for a complete dice)

Disk, source and hex code files, order code **040248-11** or Free Download

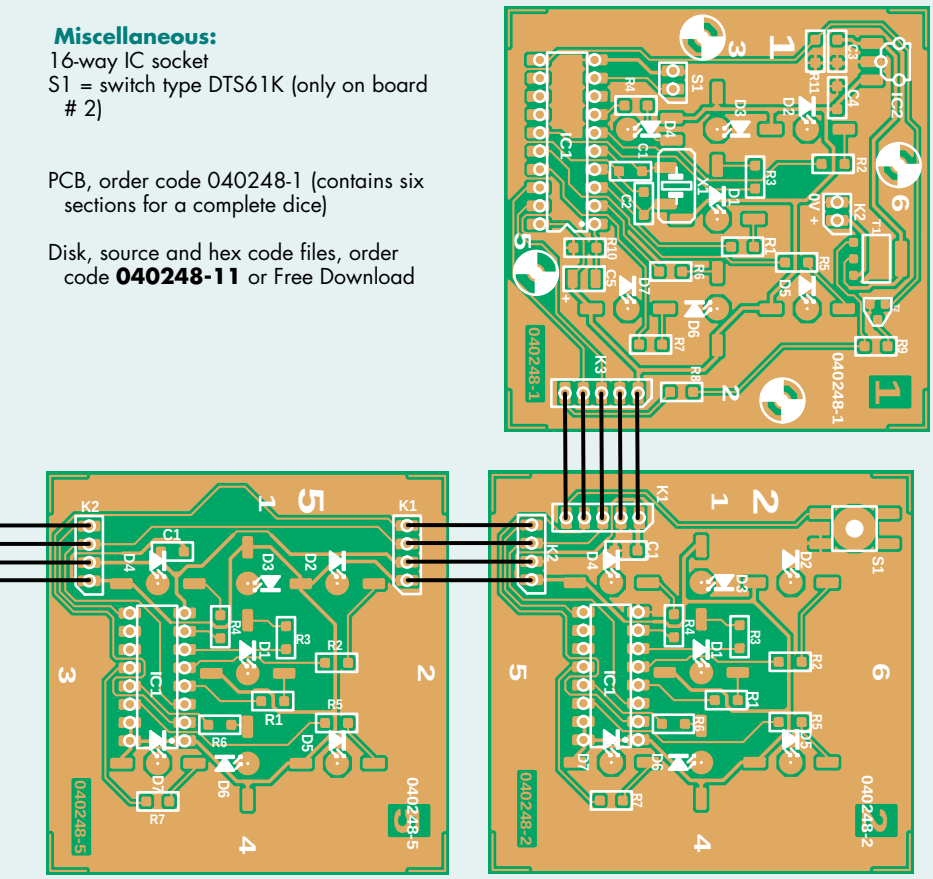


Figure 4. Component layout. Four of the six boards are almost identical.

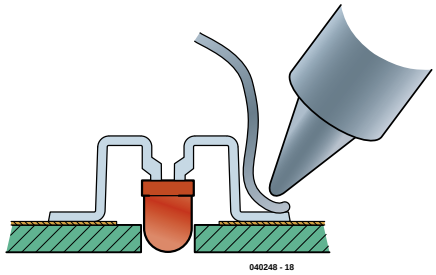


Figure 5. Bend the leads of the LEDs according to this example.

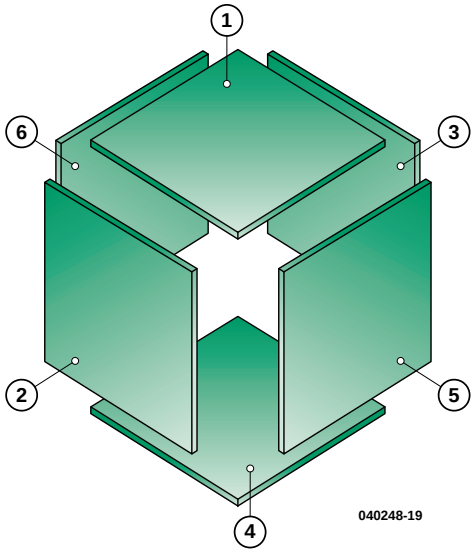
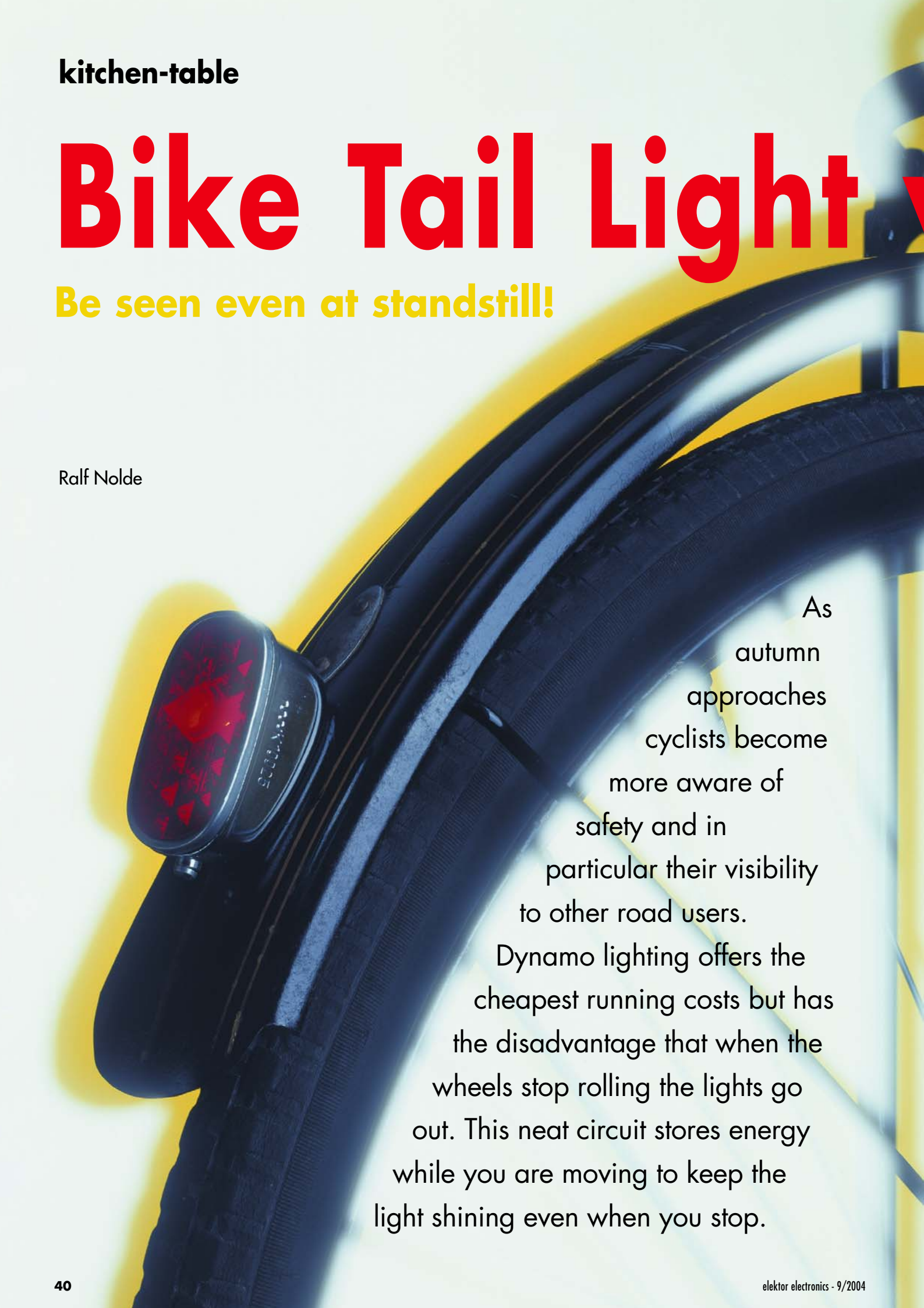


Figure 6. The six boards make up a cube.

Bike Tail Light

Be seen even at standstill!

Ralf Nolde



As autumn approaches cyclists become more aware of safety and in particular their visibility to other road users. Dynamo lighting offers the cheapest running costs but has the disadvantage that when the wheels stop rolling the lights go out. This neat circuit stores energy while you are moving to keep the light shining even when you stop.

with Standlight

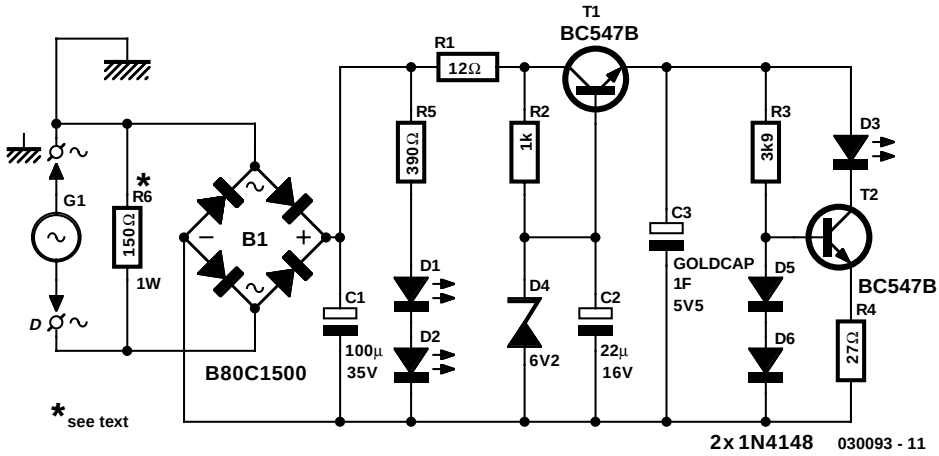


Figure 1. The circuit looks similar to a stabilised mains power supply.

It is not too difficult to construct a simple and reliable rear light unit for a cycle dynamo using high visibility LEDs. They have a much longer lifespan than filament lamps and this design uses a high value capacitor to store enough energy to keep the lamp burning for a few minutes after the cycle has come to a standstill.

The circuit

The circuit diagram for the rear light is shown in **Figure 1**. A typical cycle dynamo will only have a single output terminal; the return path is made through the dynamo body and cycle frame. A connection to the frame will form the earth input while the dynamo output is wired to the D input on the circuit. Rectifier B1 produces a full-wave rectified DC voltage and capacitor C1 provides some smoothing.

The Goldcap capacitor C3 is charged from the rectified dynamo output through resistor R1 and transistor T1. The zener diode D4 fixes the base of T1 at 6.2 V and its base-emitter drop of 0.7 V ensures that the voltage output at the emitter does not exceed the 5.5 V maximum voltage rating of C3. Capacitor C2 smoothes the voltage reference and resistor R1 limits the charging current through the transistor.

The constant voltage produced by T1 is used to charge C3 and also provide energy to light LED D3 when the bike is moving. D3 is an 8 mm diameter ultra bright red LED with a clear lens and a supply current of 26 mA. The 'F' suffix indicates that this model has a high luminous intensity in this range of LEDs manufactured by Kingbright (available from Maplin). The LED is driven by a constant current source formed by T2 and this ensures that its brightness is largely independent of the voltage stored on C3. Diodes D5 and D6 fix the base of T2 at approximately 1.4 V which in turn defines the voltage drop across R4 at 0.7 V because it is equal to 1.4 V minus the base-emitter conduction voltage of T2 ($V_{BE} \approx 0.7$ V). With the voltage across R4 fixed a quick application of Ohms law indicates that the current through R4 (and therefore D3) is constant at around 26 mA.

With the bike at standstill LED D3 is powered from energy stored in C3. Using the components specified D3 remains lit for at least two minutes before any decrease in brightness is noticeable. This afterglow period is defined by resistor R4, increasing its value to 33 Ω or 39 Ω will prolong the afterglow at the expense of reduced LED brightness. The light intensity of D3 is much higher than a conventional filament lamp but has a smaller beam angle. The two 'driving' LEDs D1 and D2 are powered directly from the dynamo output and are positioned either side of D3 to increase the effective beam angle of the rear light assembly when the cycle is moving.

Current consumption of the circuit is approximately 45 mA and with a dynamo output voltage of 6 V this gives a total power consumption of around 0.3 W. This figure is about one half of the power consumed by a conventional filament type rear lamp and reduces loading on the dynamo. The effect of this is that the voltage to the front lamp will be slightly increased

UK laws at a standstill?

In the UK lights on a bicycle are not required when stationary and to comply with the law the lamp should have a filament! British Standards have been updated to allow LED light sources and these have been commonly in use for a number of years but to be legal the lamp should have a filament. So a lamp with a BS approval could be illegal. The law is seldom if ever enforced on this point and rightly so because it's safety that matters. Cyclist also use flashing lights even though they are also technically illegal. Surveys indicate that most cyclists (about 90%) use battery powered lights.

The two documents concerning cycle lighting are the 1989 Road Vehicle Lighting Regulations and British Standard 6102 part 3. Any form of home-made lighting will not have BS approval. The Bike Tail Light described in this article should only be used as a secondary means of lighting which is fitted together with a BS approved system.

COMPONENTS LIST

Resistors:
R1 = 12Ω
R2 = 1kΩ
R3 = 3kΩ9
R4 = 27Ω
R5 = 390Ω
R6 = 150Ω 1W

Capacitors:
C1 = 100µF 35V radial
C2 = 22µF 16V radial
C3 = Goldcap 1F /5.5V (Conrad Electronics # 473120)

Semiconductors:
B1 = B80C1500, round case (80V iv, 1.5A)
D1,D2 = LED, red, 3mm, Kingbright L-934SRC-J (Reichelt, LED 3-3500RT)
D3 = LED super-red, 8 mm, Kingbright L-793SRC-F (Maplin # N46AT)
D4,D5 = 1N4148
D6 = zener diode 6.2V, 500mW
T1,T2 = BC547B

Miscellaneous:
Bike rear light case, Busch & Müller type 339, www.bumm.de/docu/ruecklicht1.htm (UK distributor: Amba Marketing)
PCB, available from the PCBShop

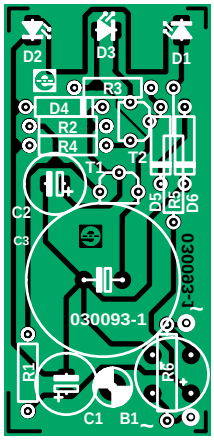


Figure 2. The small PCB is designed to fit into a standard rear light assembly.

and it will burn brighter for a given speed. This will reduce the front lamp lifespan if you habitually descend long hills at high speed. Resistor R6 is fitted as a 'dummy load' to avoid this potential problem but if you never break into a sweat while pedalling and your cycle is the type fitted with wicker baskets, you are unlikely to need this precaution and R6 may be omitted.

Construction

The circuit can be built on the single-sided circuit board. The PCB layout accepts conventional wire-ended components with some of them fitted to the underside (alternatively you can keep them all on the same side by fitting R5 over the top of D5/D6 and R6 under rectifier B1).

The PCB is quite small but this should not pose any problems when mounting the components. It is important to double check that all polarised components (diodes, transistors and electrolytic capacitors) are fitted the correct way round before power is applied to the circuit.

A 9 V battery or mains unit can be used to test the unit. The power source is connected in place of the dynamo, polarity is not important because the bridge rectifier ensures that power will always be correctly supplied to the circuit. At power-up all the LEDs should light and after a short charging time the voltage across the Goldcap can be measured (approximately 5.5 V) while the voltage drop across R4 should be less than 0.75 V.

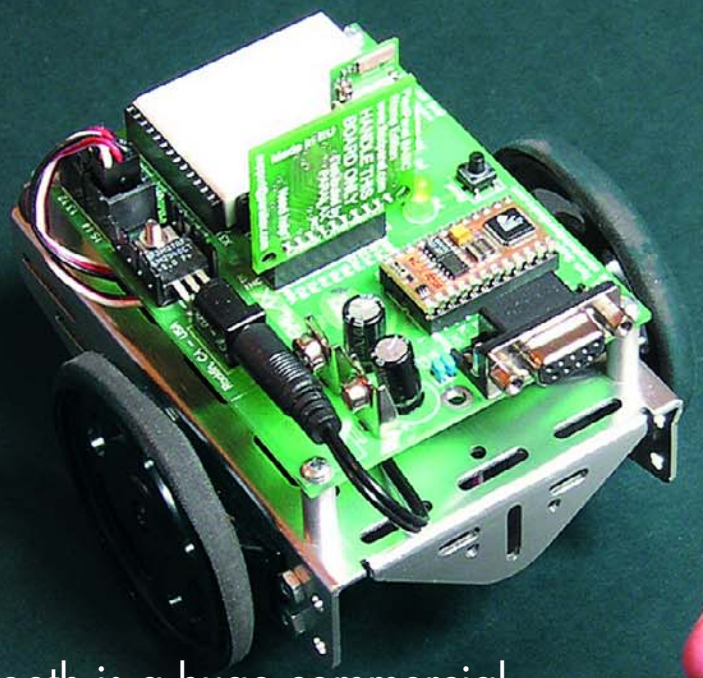
Once the circuit has been tested, mask off the LEDs with tape and give the whole unit a few coats of spray lacquer to protect it from the effects of the weather. Once the lacquer is dry the unit can be mounted in the taillight housing. The PCB is dimensioned to fit in a housing type 339 made by the company Busche & Müller and distributed on the UK by Amba Marketing. It can also be adapted to fit into any similar rear light housing.

For the housing specified, the PCB is fitted behind the reflector part and fixed with a small (rust proof) self-tapping screw. Cycles are subject to surprisingly high levels of shock and vibration so it is a wise precaution to support all the major components like capacitors transistors and LEDs with an application of hot-melt glue after the circuit has been fitted and wired to the dynamo. Lastly fit the rear light lens and ensure that the LED is correctly aligned with the reflector lens opening.

(030093-1)

Bluetooth Remot

Richard Hoptroff



Bluetooth is a huge commercial success but so far hobbyists and lab workers have been barred from access to ready-made modules for this wonderful new medium. The reason is simple: manufacturers like to treat their products as black boxes and are only interested in their 100-k/day production lines in the Far East or China. We oppose this attitude by publishing this article, written using the motto: Bluetooth modules for everyone! A tracking robot does the trick.

te Control from your PDA or mobile phone

Flick through any electronics magazine – professional or hobbyist – and you will see a wide range of single-board computers and microcontroller boards. For many applications, they make product development so much simpler than it used to be, say, 5 or 10 years ago. Attach a few auxiliary components and a control panel, write the computer program, and you're finished.

What makes the process easier is that the computer board is programmable, so one off-the-shelf component can be applied to many tasks. Could this concept be taken even further? A few auxiliary components will always be needed in any product, but what about the control panel (user interface)? Couldn't an off-the-shelf programmable component be made to replace custom control panels on electronic devices like PDAs, GSM, but also custom-made microcontroller systems?

The 'FlexiPanel BASIC Stamp Edition' module from FlexiPanel Ltd. could provide the answer. In this article we'll concentrate on the Bluetooth version of Flexipanel, see the **Device Pinout inset**. Using Bluetooth radio (at 2.4 GHz), it asks a remote device within range — say, a mobile phone, a notebook PC, PDA or another handheld computer — to create the required control panel (or, if you like, 'GUI') on its display. The module has a Class-1 radio, so the remote device can be up to 100 m away. The module operates at TTL levels, and we are informed that a standalone RS232 device will soon also be in production.

A user within radio range of the Flexipanel-Bluetooth module may connect to the appliance at any time using any Bluetooth-enabled device. The device will display the required control panel, but the panel's appearance may vary according to the remote device used. Some examples will be shown later.

The software on the remote devices is the same for each application and does not require customization or re-installation. It is freely downloadable from www.flexipanel.com. At the time of writing, Pocket PCs, Windows PCs, and Smartphones (e.g. SPV E200 from Orange) software are supported. Software for Palm Operating System and Java phones supporting the JABWT standard (e.g. Nokia 6600 and Sony Ericsson P900) has also been released.

Projects with Bluetooth control

Let's not get carried away by new-fangled technology like Bluetooth-enabled GSM phones and PDAs. Using an example we will demonstrate that applications can be developed for the FlexiPanel-Bluetooth combination that are simple enough to be tackled by relative beginners. The example, a simple tracking robot, employs a specially adapted version of the **Elektor Electronics Board of Education** originally featured in the September 1999 issue and the world-famous **Parallax BS2p BASIC Stamp**. Parallax Inc., who supply the Basic Stamp, also distribute the FlexiPanel module through their authorised dealers (in the UK: Milford Instruments).

In case you didn't know, the BASIC Stamp can be programmed using the BASIC programming language from any PC computer using a serial cable. The same link is used to program the control panel into the FlexiPanel-Bluetooth module used as part of the project: a robot controller with route tracking.

The BASIC programs and FlexiPanel designer data files used in this and two more projects are available from *Elektor Electronics* as free software downloads. The other two projects are an access system and a temperature logger, they are not discussed in this article.

Modified BoE

The circuit diagram of the modified BoE (Board of Education) is shown in **Figure 1**. The 'brains' are a BS2p (BASIC Stamp 2) module which is plugged into the 24-way socket. Like its 1999 predecessor, this BoE has an RS232 connector (K2) and a prototyping area enclosed between K7, K3, K6, K8, K4 and K5. We'll use the area later to fit a couple of components the robot needs to be able to tell you its whereabouts.

The Flexipanel Bluetooth module is plugged straight onto a dedicated socket, K9.

The BoE has its own voltage regulator, IC2, and when used on its own (for example, during programming sessions) can be powered from a 9-volt battery. The programming utilities and the BASIC editor are available from Parallax Inc., just look for Board of Education on their website www.parallax.com and you'll find a mass of (free) information.

The component mounting plan of the new BoE is given in **Figure 2**. Hoorays and applause at this point because the board is **single-sided**.

Tracking robot

If your friends sniff at yet another buggy-style little robot, tell them that this remote controller differs from many others in being able to *send information back to the handheld device using data over a radio link*. By using an electronic compass mounted on the robot, a route trace is recorded and reported back to the handheld unit.

Figure 3 shows what to add to the BoE to make it suitable for our experiment. First, there is the combined FlexiPanel-Bluetooth module hooked up to the BS2p by five lines. The two units employ bidirectional serial communication with handshaking. As you will have surmised, the FlexiPanel is also a

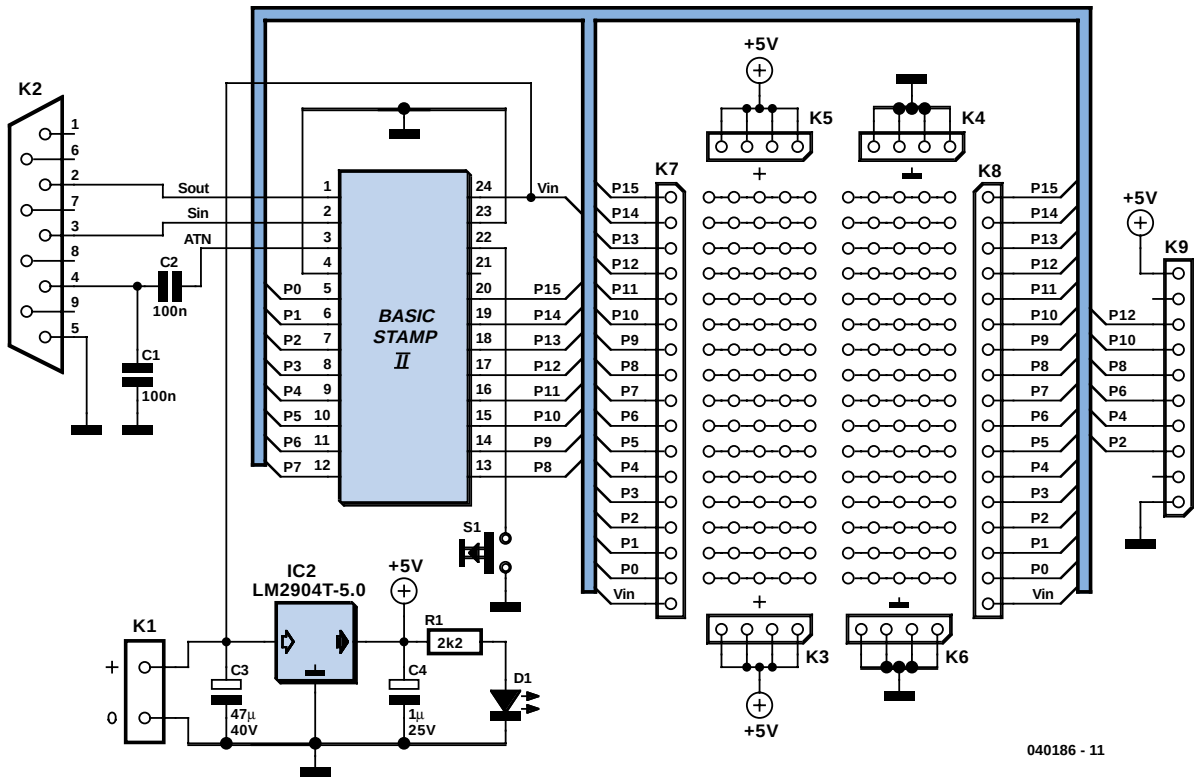


Figure 1. Circuit diagram of the Board of Education (BoE), specially adapted for the Flexipanel-Bluetooth module from Parallax.

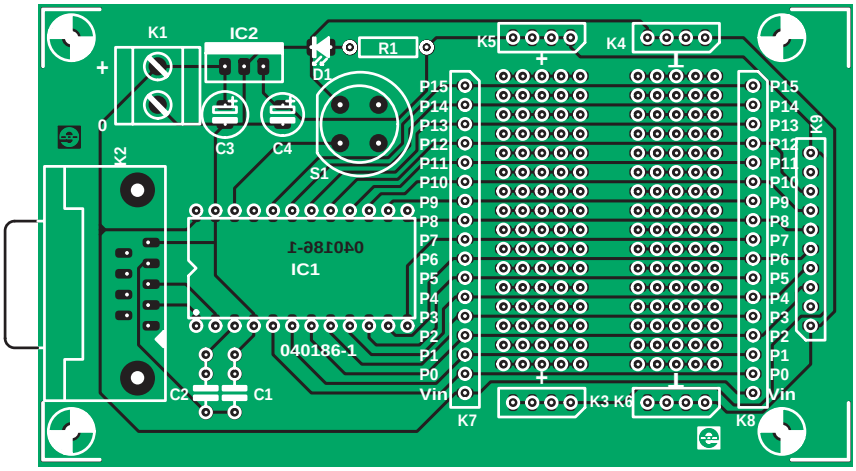


Figure 2. PCB design for the modified BoE.

COMPONENTS LIST

Resistors:
R1 = 2kΩ2

Capacitors:
C1,C2 = 100nF
C3 = 47µF 40V radial
C4 = 1µF 25V radial

Semiconductors:
D1 = LED, red, low current
IC1 = Basic Stamp (BS2, BS2sx, BS2e or BS2p) (Parallax Inc, Milford Instruments)
IC2 = LM2940T-5.0

Miscellaneous:
K1 = 2-way PCB terminal block, lead pitch 5mm
K2 = 9-way sub- D socket (female) angled pins, PCB mount
K7,K8 = 17-way SIL connector (header or socket)
K9 = 10-way SIL socket
S1 = pushbutton, 1 make contact, PCB mount, e.g., D6R

microcontroller system (and an intelligent one, too)! The BS2p runs software capable of sending commands that request or modify FlexiPanel values, content or status information. Moreover, FlexiPanel can request BS2p attention when a client device has changed a control via Bluetooth. This is done using a kind of interrupt conveyed via the Data line which in our case is monitored by an LED.

The electronic compass module type CMPS03 is an I²C device from Devantec. It is available from, among others, Milford Instruments.

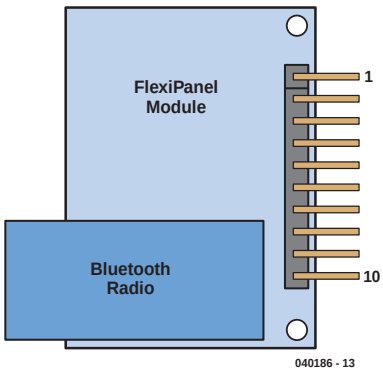
The BoE with its extension circuitry crammed in the prototyping area is mounted on the BoE-Bot robot superstructure available from Parallax Inc. This has motorized wheels which may be controlled by pulsewidth modulation direct from the BASIC Stamp as indicated in Figure 3. The tracking robot, ready to start on its journey, is shown in the introductory photograph. Note that the photo shows the Parallax BoE.

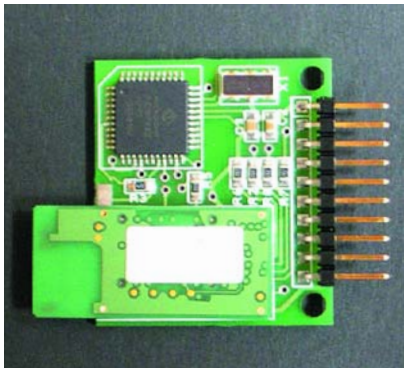
FlexiPanel and BS2p programming

Using *FlexiPanel Designer*, a programming utility supplied by Hoptroff, a user interface is created containing:
– compass bearing display;

Device Pinout

Pin	Name	Purpose
1	Vss	Connect to 0 V
2		Not connected
3	RxD	Serial data input from BASIC Stamp for serout operations
4	TxD	Serial data output to BASIC Stamp for serin operations
5	RTS	Serial flow control output to BASIC Stamp for serout operations
6	CTS	Serial flow control input from BASIC Stamp for serin operations
7	Mod	See text.
8	Data	Data output high when a control has been updated by a FlexiPanel client.
9		Not connected
10	Vdd	Connect to +5 V.





Care must be taken to insert the module into the correct side of the AppMod slot and in the correct orientation. Make sure Vss connects to Vss and Vdd connects to Vdd (not Vin!). Failure to do so may damage the module.

– latching pushbuttons for stop, forward, reverse, left and right;
– a table showing the route traced by the robot.

Flexipanel Designer generates a program for PBasic that allows the FlexiPanel hardware to be programmed. Using the PC and Basic Stamp Editor, this piece of software is downloaded onto the Stamp where it is executed.
It may happen that FlexiPanel has to be reset first, for example, when an earlier attempt at programming went wrong. In that case, the circuit supply voltage has to be switched off and on again — the reset button on the BoE having an effect on the Stamp only and **not** on the FlexiPanel! When the circuit is switched on, the programming will recommence automatically — the program still being available in EEPROM, it does not have to be loaded again from the PC. It should be noted that FlexiPanel needs about 10 seconds to boot so you can stir your tea or coffee before programming actually commences. To have at least an indication of what's happening during the boot-up phase, a low-current LED may be connected between the Data line and ground, not forgetting a 1-k resistor — see Figure 3. After switching on the supply or after a soft reset of the module (which happens automatically after programming), the LED will light up for a few seconds and then go out once booting is finished. The LED will also light briefly when FlexiPanel receives a command over Bluetooth (interrupt request).
The result of the using *FlexiPanel Designer* may be seen in the **What do I with it** inset: simple buttons to press on a pocket PC or GSM phone, and a map returned by the robot telling you where it went!
When the user interface has been programmed into the FlexiPanel module, the BASIC Stamp is loaded with the runtime program shown in **Listing 1**.

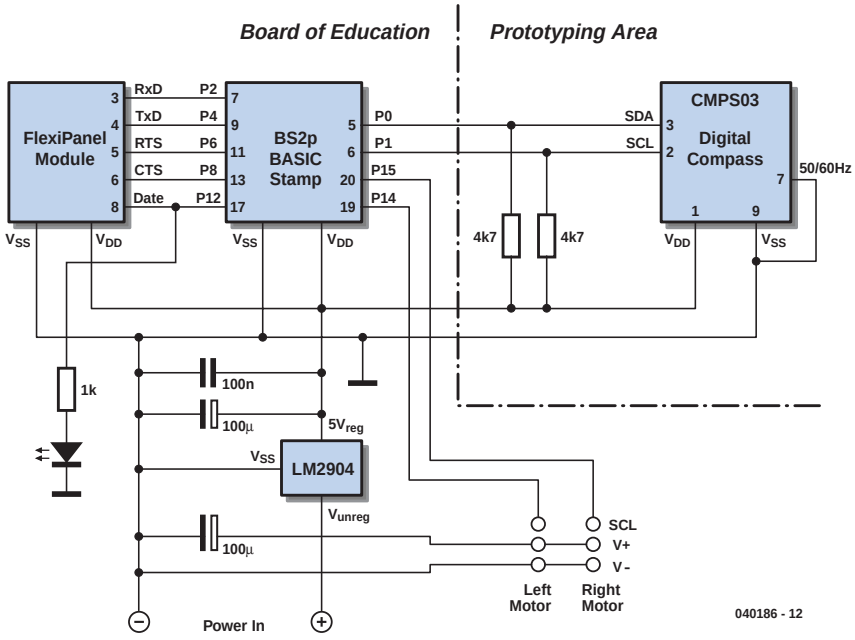
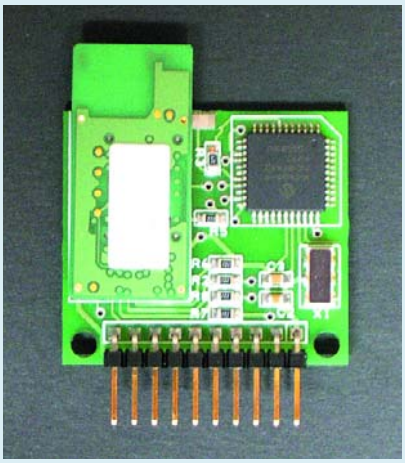


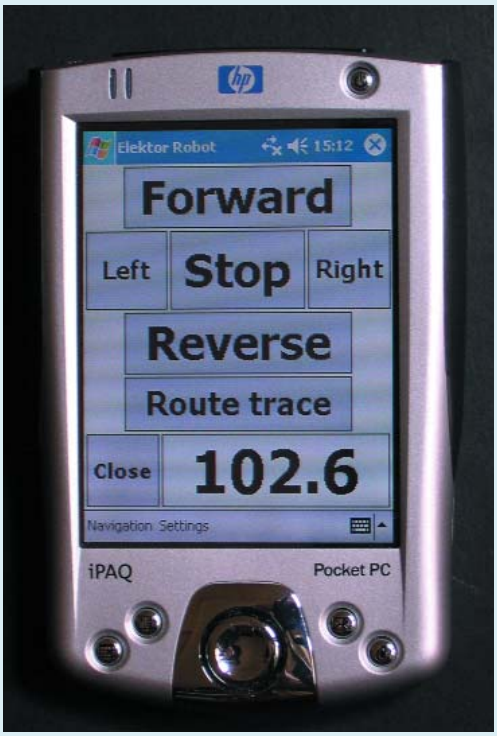
Figure 3. The add-on circuitry for the Tracking Robot consists of an electronic compass module.

What can I do with it?

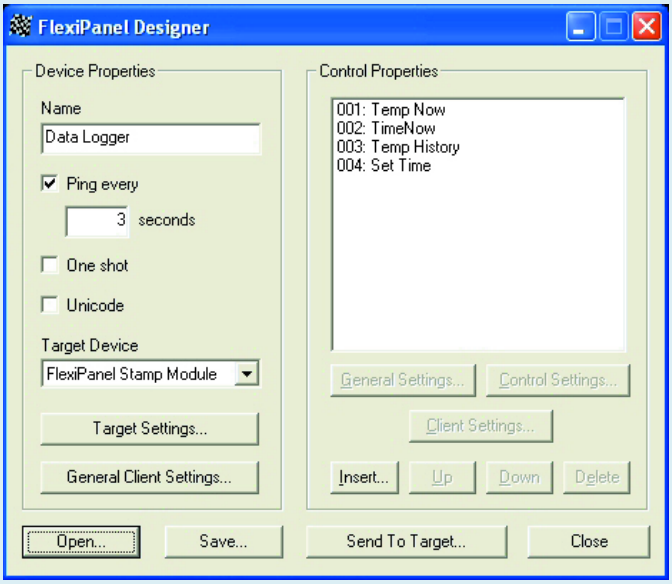
The purpose of this inset is to provide incentives to developing projects using the FlexiPanel-Bluetooth module from Parallax. Have look what can be done with it! The project documentation and software are available as free downloads.



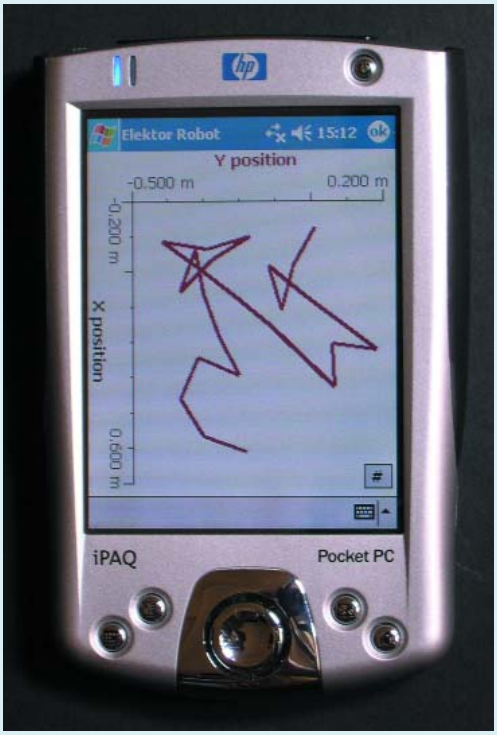
FlexiPanel with the piggy-back mounted Bluetooth radio sub-board. You have the pin-out and datasheets — nothing to stop you from hooking up a microcontroller.



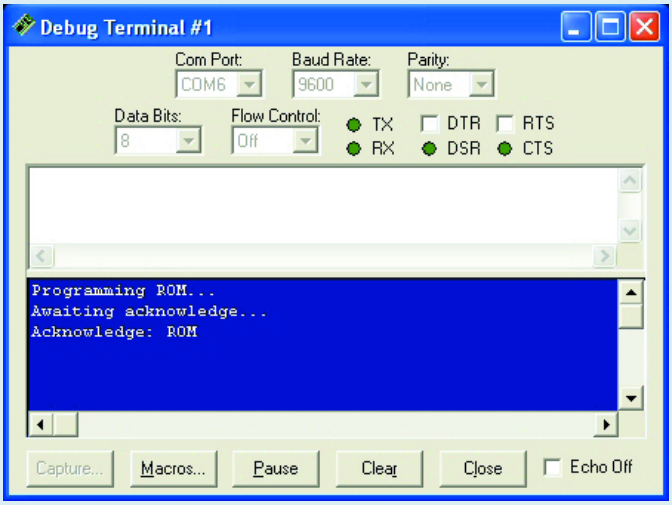
Tracking Robot user interface on a Pocket PC.



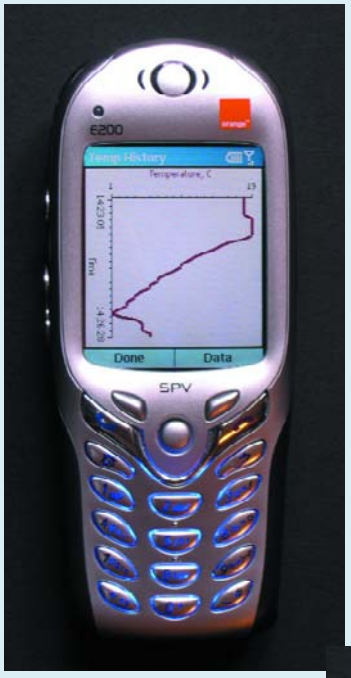
FlexiPanel Designer for a Data Logger project. Design your own control buttons as they appear on Bluetooth devices.



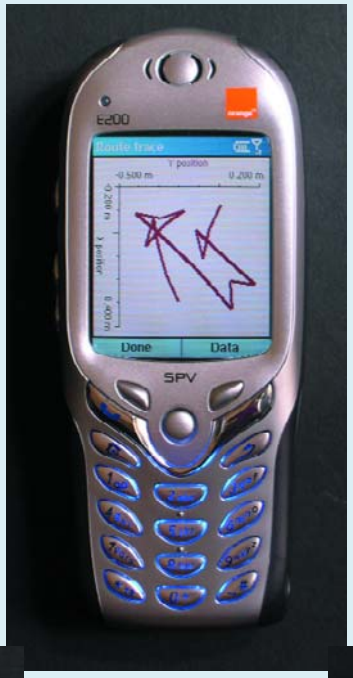
I'm sure we missed a turn somewhere, dear! The Tracking Robot route trace displayed on a Pocket PC.



BASIC Stamp Editor busy programming the FlexiPanel module (old version, now replaced by drag 'n drop).



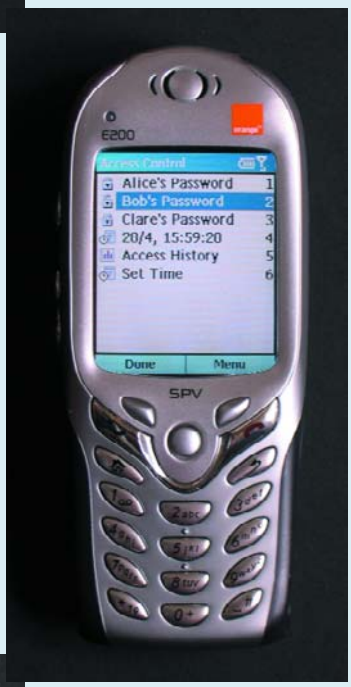
Temperature history on a Smartphone.



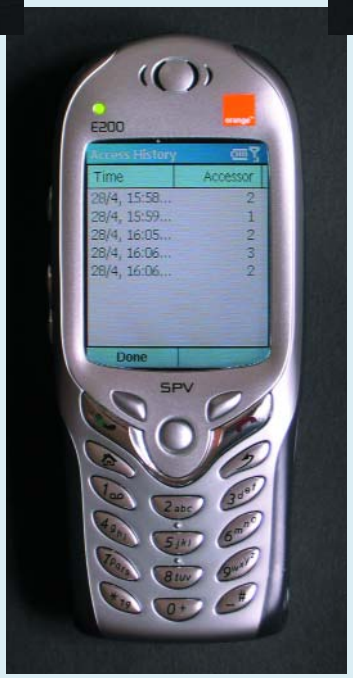
We're lost, aren't we? Tracking Robot route trace displayed on a Smartphone.

Temperature Logger graphic user interface on a Smartphone.

Access Controller graphic user interface on a Smartphone.



Tracking Robot graphic user interface on a Smartphone.



Access Controller log displayed on a Smartphone.

A Bluetooth Protocol

Bluetooth is a 2.4GHz digital radio communication protocol developed and licensed by Ericsson. Serving the 'personal area network', Bluetooth devices can come and go ad hoc. In contrast, the WiFi protocol, operating at the same frequency, is more suited to longer-term wireless infrastructure, with each individual node needing to be assigned a fixed IP (internet protocol) address.

Thanks to Bluetooth headsets, Bluetooth is now solidly entrenched in the mobile phone market. Intel intends to incorporate Bluetooth into its Centrino 2 chipset, to be launched in Autumn 2004. Not only will this allow PCs to connect wirelessly to printers, etc, but it will boost the growth of VoIP (voice over internet protocol), i.e. phone calls over the internet.

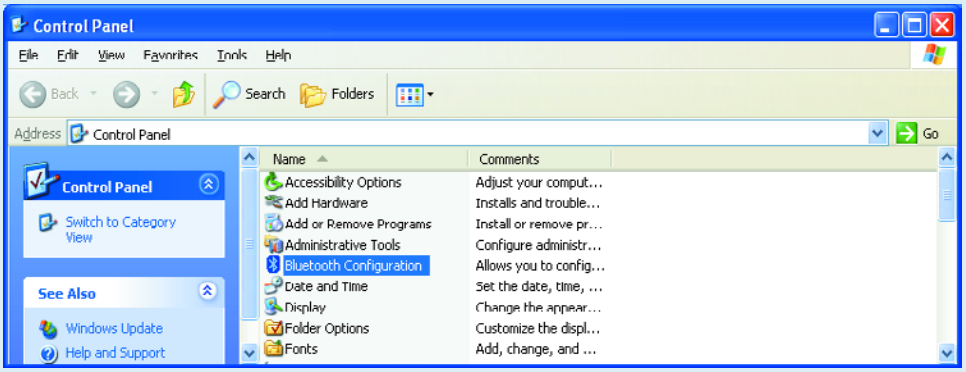
The Bluetooth standard provides interfaces for a wide range of communications protocols, from a simple serial port to audio. Like many higher-level protocols such as OBEX file exchange, FlexiPanel sits on top of the serial port emulation layer of the Bluetooth protocol stack. It is not part of the 'official' Bluetooth standard. However, the standard is relatively open in that anyone is free to create software for remote devices, and product-side components such as the FlexiPanel module are manufactured under license, just like any Bluetooth radio module. The first FlexiPanel products were software libraries to provide remote control for Windows applications and high-end embedded systems.

From the electronic product's perspective, the FlexiPanel module is a peripheral providing graphical user interface services. It maintains a list of the controls required by the product, and the current state of the controls. The product can update a control at any time and if a user modifies a control, the product is notified!

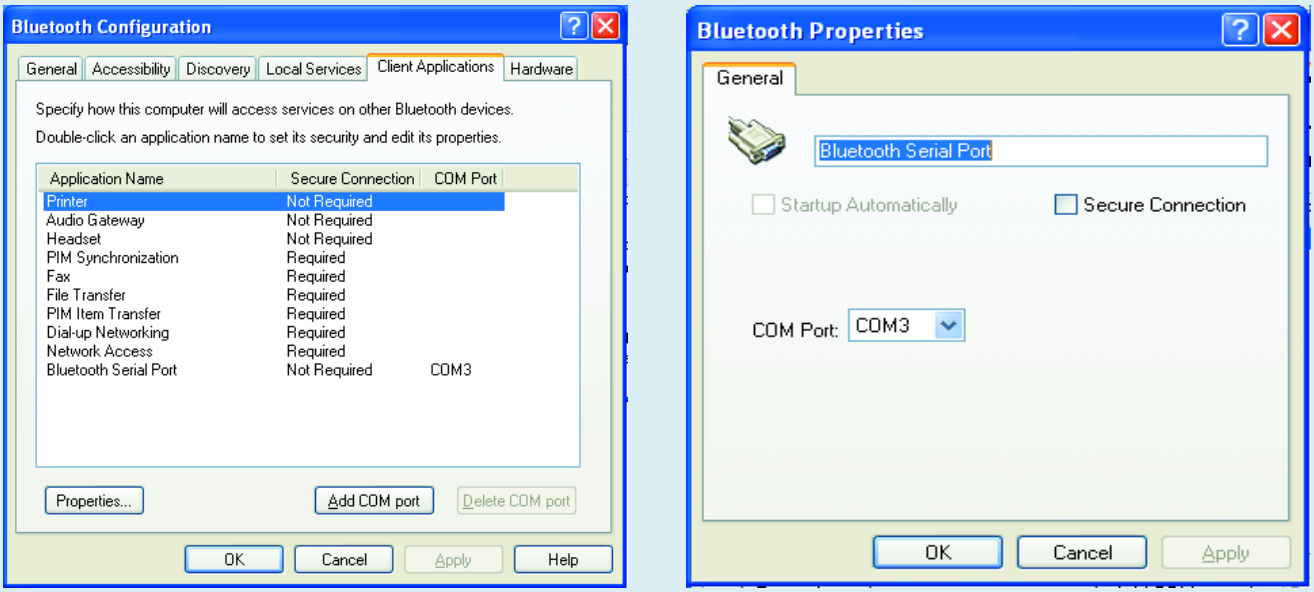
Installing a Bluetooth adapter

Desktop PCs will typically employ external Bluetooth adapters to communicate with devices like FlexiPanel. Usually, the installation of such adapters is a breeze using the supplied software, however in the case of the FlexiPanel some fine tuning is required.

For these adapters a series of COM ports is installed that look after the link to other Bluetooth devices. These links are normally protected against non-authorised access to the PC by the installation software. However, FlexiPanel does not employ a protected link and that's why you need to tweak the PC's COM port settings. You start from the Control Panel where the Bluetooth configuration may be found.



Double-clicking on this option takes you to the next window (tab: Client Applications)



Inspect the COM port properties and make sure Secure Connection is unchecked.

For these and other niggling details on Bluetooth see the FAQ at
www.flexipanel.com/comports/index.htm

Listing 1. BS2p code for Tracking Robot (extract)

```
BackUp:
         PULSOUT lPort, lMaxZ
         PULSOUT rPort, rMaxZ
         PAUSE 20
         FwRvSp = 2
         GOSUB CheckCompass
         IF DataPin = 1 THEN ReadControls
         GOTO BackUp

CheckCompass:
         ' only check every 50 pulses
         CmpCount = CmpCount - 1
         IF CmpCount > 0 THEN GoBack
         CmpCount = 50

         ' Get compass direction in binary radians and in tenths of a degree
         I2CIN  SerPt, CmpIn, 1, [brad, degs.HIGHBYTE, degs.LOWBYTE]

         ' Send degrees value to bearing control (code generated by FlexiPanel Designer)
         SEROUT TxPin\CTSPin, BaudM, [SetData, ID_Bearing, degs.LOWBYTE, degs.HIGHBYTE, 0, 0]

         ' Calculate position with Send binary radians value to bearing control
         ' (code generated by FlexiPanel Designer & cut'n'pasted)

         IF FwRvSp = 1 THEN
           xloc = xloc + COS( brad )
           yloc = yloc + SIN( brad )
         ELSEIF FwRvSp = 2 THEN
           xloc = xloc - COS( brad )
           yloc = yloc - SIN( brad )
         ENDIF

         ' if moving, send to trace
         IF NOT FwRvSp = 0 THEN
           SEROUT TxPin\CTSPin, BaudM, [AddRow, ID_Route_trace, yloc.LOWBYTE, yloc.HIGHBYTE,
                  xloc.LOWBYTE, xloc.HIGHBYTE ]
         ENDIF

         ' return to motor control
GoBack:
         RETURN
```

After initialization, the program tests to see what kind of motor control pulse it is supposed to supply. Then it reads the compass and writes the bearing to the bearing display and the route tracker.

More about the project software

To get started with this project you will need the following:

1. Windows 2000 or later
2. FlexiPanel Software Development Kit (SDK)
3. Basic Stamp Editor (BSE)
4. FlexiPanel

Some notes: the FlexiPanel Software Development Kit SDK (previously known as FlexiPanel BASIC Stamp Developer's Kit) only works on recent versions of Windows.
The SDK allows the GUI to be designed on a PC, i.e. you decide on the controls necessary for the final product, and their 'look' as they appear on the Bluetooth device, see 'FlexiPanel Designer' above.
Regarding the BSE, only the Bs2, Bs2sx, Bs2e and BS2p are supported. The –p version is preferred because of its speed and memory size. Note that the 40-pin Bs2p40 will not fit on any BoE as only a 24-pin socket is available.
A version of FlexiPanel for Windows 95/98 is available as a free download, as well as versions for MS Smartphone, Pocket PC, Java JABWT devices and Palm OS.

Closing notes on the tracking robot project

The BoE-Bot is powered by four AA batteries which, in practice, will only last a few tens of minutes before they are exhausted or their voltages start dropping to levels where the electronics start 'browning-out'.
The FlexiPanel module can generate a signal indicating when a remote unit is connected. The robot could automatically halt if it went out of range of the remote unit.
The operation of the electronic compass is significantly influenced by the surrounding metal and direct currents. These effects may be counteracted by local calibration of the compass.

(040186-1)

Web pointers

Parallax: www.parallax.com
FlexiPanel: www.flexipanel.com
Milford Instruments: www.milinst.demon.co.uk

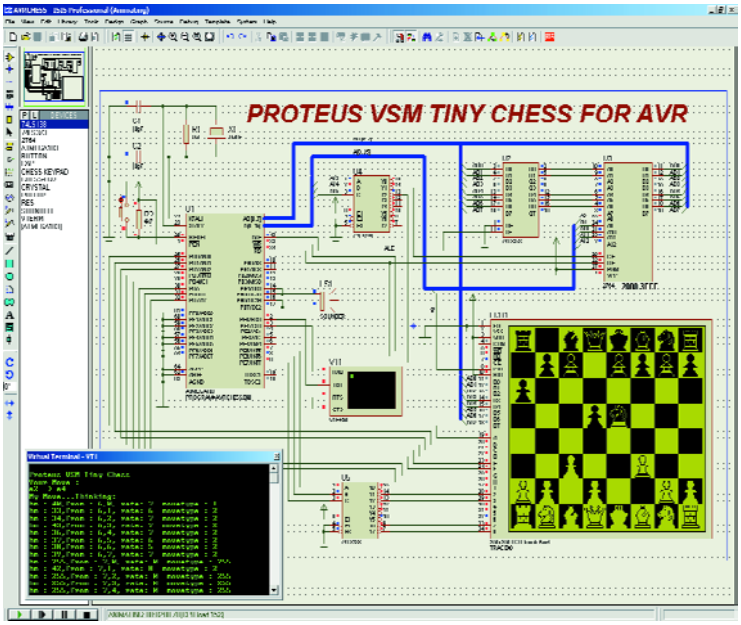
Free Downloads

Basic stamp (.bsp) and Flexipanel (.FxP) programs for Tracking Robot, Datalogger and Access Controller projects. Full project documentation (.doc). File number: 040186-11.zip
PCB layout in PDF format. File number: 040186-1.zip
www.elektor-electronics.co.uk/dl/dl.htm, select month of publication.

Proteus VSM

Also simulates microcontrollers!

David Daamen



Schematic drawing, PCB designing and off you go testing the design. These days we do everything on a PC. Simulation programs have now reached a level of sophistication that includes co-simulation of a microcontroller in your design. Proteus VSM from Labcenter Electronics is an extensive software bundle offering professional features at an affordable price.

Proteus Virtual System modelling (VSM) is software that allows digital and analogue circuits, or a mixture of the two, to be simulated on a PC. Of course, it all starts with circuit diagram entry ('schematic capture'). Here, ISIS schematic capture is used. From the same supplier comes a separate utility, ARES, for the PCB design phase, providing a seamless link with ISIS. Together, ISIS and ARES form the traditional combination that makes 'putting a circuit onto PCB' very easy indeed. Naturally we found features like connectivity verification ('electrical rule and connectivity check'), an automatic component placer and an automatic track router. So far, nothing unusual as other software products may be found with roughly the same functionality.

In the case of Proteus VSM, it's the simulation part that offers a number of unusual features, including one we've not seen before in affordable design/simulation packages: microcontrollers may be included in the circuit. That, in itself, is possible with other design software, but Proteus VSM is actually capable of simulating the code executed by the micro! Again, you may argue that even that is not unique but wait, there's another novelty. During simulation, you can work interactively with the circuit: switches and potentiometers may be operated 'as if for real' and the status of LEDs or information on LCDs is immediately visible. The feature is fast, too, as it is perfectly possible to make periodic signals with frequencies in the audio range audible via the PC's soundcard. A complete graphic LCD touch screen hooked up to a microcontroller is not only refreshed during simula-

tion, but may also be controlled using the mouse. It all happens virtually in real-time. On a 300-MHz Pentium II you can simulate a standard 8051 system ticking at 12 MHz.

The simulator

The core of the simulator is called ProSPICE, a combination of an analogue simulator and a fast event-driven digital simulator, allowing a mix of analogue and digital electronics (a 'mixed mode' circuit) to be simulated without problems. The advantage of SPICE is that component models supplied by manufacturers may be added at the drop of a hat. Incidentally, Proteus VSM comes with about 6,000 SPICE models.

The simulation also employs 'animated models': graphic animations whose appearance can change during simulation, for example, to indicate the current flow direction, or showing that a lamp actually lights up. It is also possible to define your own (animated) models, in principle, without the need to program them!

A documented 'software developers kit' (SDK) is supplied allowing models to be created in the form of DLL files.

Measuring

Virtual testing implies virtual measurement — not just using simple volt and ammeters, but also more advanced instruments like an oscilloscope, a function generator, a pattern generator, a counter and an asynchronous terminal. As far as operation and options are concerned, these are simply 'instruments' as you may see them on the test bench. A useful 'extra' is the ability to display the logic level of any connection or junction in the circuit in real time using a coloured dot.
More advanced measurements like graph plotting or frequency response measurement, noise and distortion analysis, are available in the utility 'Advanced Simulation Option'.

Microcontroller co-simulation

VSM is capable of linking two simulations: one of software executed 'inside' a microprocessor, the other, of the complete electronics circuit around the micro. For example, the act of a processor writing code to a port results in corresponding logic level changes in the circuit. And the other way around: if a (sub-)circuit changes a logic level then the simulated program will notice it.

VSM's included models not just support simple I/O ports but also interrupts, timers, USARTs etc. — depending of course on the processor being simulated. It is even possible to include multiple processors in the circuit — it's just a matter of drawing a few lines to connect the relevant chips!
At the time of writing, Proteus VSM versions are available for the BASIC Stamp, PIC, AVR, HC11 and 8051 processors.

Debugging

Because the design of a microprocessor or microcontroller circuit is hardly ever finished and approved in one go, the process of debugging or 'error elimination' remains essential. Not surprisingly, Proteus VSM gives great attention to the subject. In particular the single-stepping feature will be welcomed by designers. Just as with the software debugger you normally use, code for the processor may be executed line by line. However, this time you can observe the results of the code execution on the whole circuit rather than the micro alone. Depending on the processor family selected and available programming tools it is even possible to 'step' through code written in higher programming languages like C.

Proteus VSM is highly recommended for designers frequently working on circuits containing digital as well as analogue electronics.

(040043-1)

The price of Proteus VSM is dependent on processor family, optional extensions and commercial or educational use. For the latest price info, contact

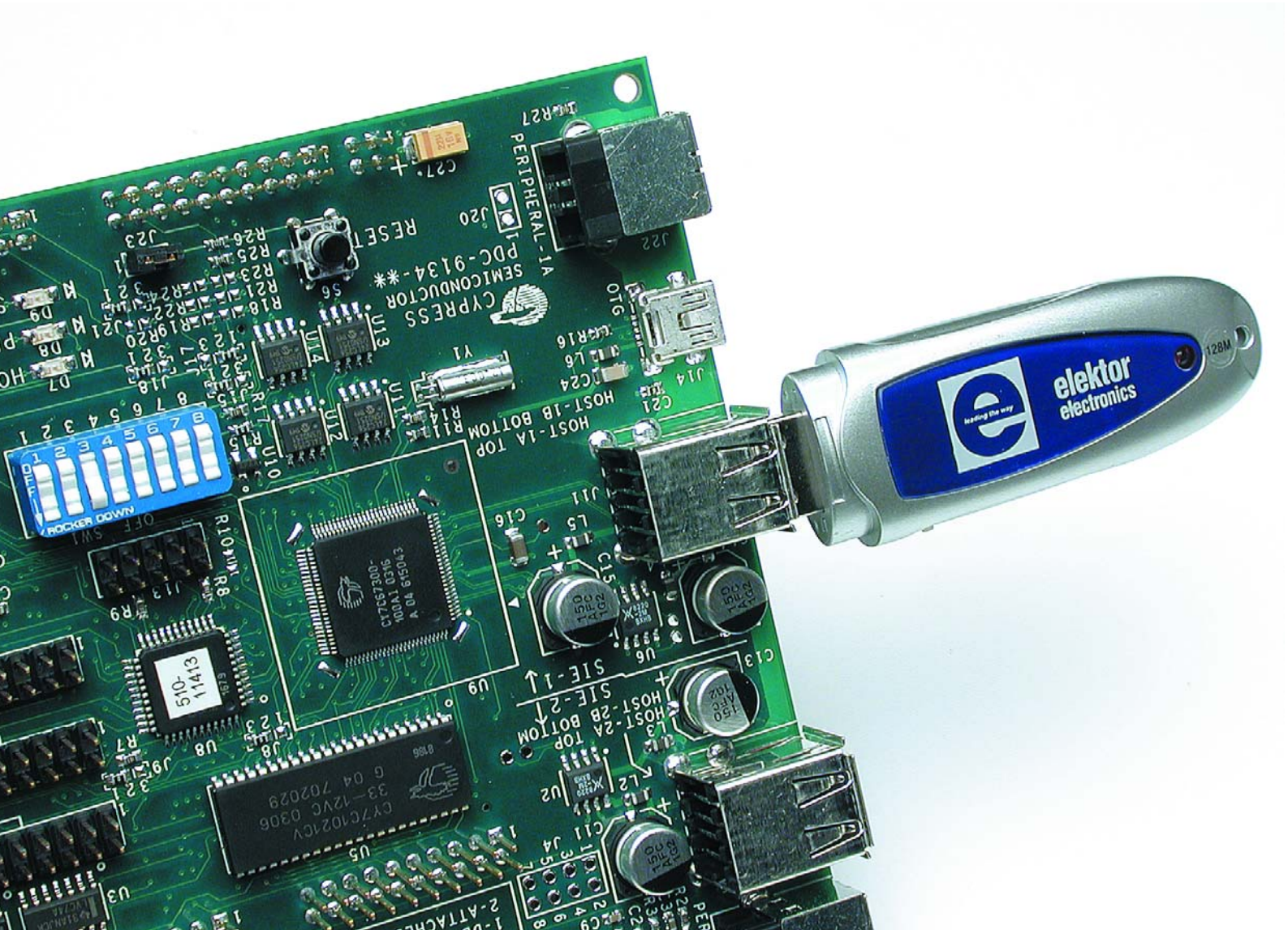
Labcenter Electronics,
53-55 Main Street,
Grassington BD23 5AA, UK.
Tel. (+44) 1756 753440,
fax (+44) 1756 752857.
www.labcenter.co.uk,
info@labcenter.co.uk

USB Embedded H

for removable mass storage devices

Mark Schultz
Senior Applications Engineer - Cypress Semiconductor

Till now USB has been available only on personal computers, leaving large, untapped potential in the area of embedded systems. New embedded host controllers from Cypress herald a new era.



OST CONTROLLER

An embedded system is defined as hardware and firmware — either stand-alone or part of a larger system — usually with some sort of operating system. The operating system can be Windows CE, VxWorks or a more simple system consisting of 'home made' code. Using the above definition, it can be said that any electronic device that has a processor has the potential to be an embedded USB host.
Designs that incorporate a USB Host controller have a distinct advantage over traditional designs in the sense that they can now host any type of USB device. The obvious advantage here is that storage space for a design can be dynamically added or removed at any time. Consider the case of an MP-3 player that contains a USB Host. MP-3 files can be easily downloaded via a USB flash drive or any other media that includes USB support. Many other types of applications should be able to realize similar performance with the addition of a Host USB solution, such as the ability to perform field upgrades or download critical data to a non-networked system.
Existing solutions for expansion of embedded systems usually involve a laptop or some other type of portable computing device connected to the embedded system via a serial cable. Although this is a simple and effective solution, it requires the extra PC and additional cabling. Additionally, there could be problems associated with the use of legacy ports.
Typically, a PC is a USB host. Anything else containing USB is a peripheral. As an example, take the case of a digital camera. A digital camera has traditionally been a USB peripheral. To print an image to a USB based printer, a PC is first used to upload the images from the camera to the PC and then to download them to a printer. If the digital camera had USB host capabilities, it could download the images directly to the printer.
Of course, such functionality comes with a price. A USB Host is responsible for a number of tasks including enumeration of devices, task scheduling and bandwidth allocation. Fortunately, there will be a code framework that will handle these tasks for us.
This article will outline some of the basics required for interfacing a USB Host controller to a removable mass storage device. A short primer on the use of SCSI commands and how they relate to file systems will be presented as well as a quick overview of the FAT file system. This will be followed by a design example.

Mass storage basics

Communication via the USB interface uses SCSI commands embedded inside a UFI Packet. A file system is a logical structure used to track various parameters of the storage media.

FAT File System

This design will support the File Allocation Table (FAT) type of file system. In the FAT file system, there is a table that contains a number corresponding to each section or cluster on the disk. It will be necessary to use the File Allocation table to locate the individual clusters within a file since they may not all be contiguous. There are other types of file systems such as NTFS (NT File System) and UFS (Unix File System) that will not be supported by this design.

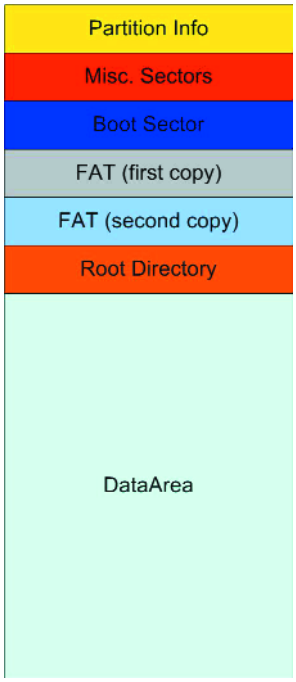


Figure 1. File structure system.

Our file system will be logically represented by a data structure called a volume. A volume contains information about the location of the File Allocation Table (FAT), root directory, and a data area that is subdivided into sectors and clusters.
All mass storage devices contain a boot record. Aside from containing code that can be used for bootstrapping the device, this area also contains other useful information about the device that can be used to fill in the volume information. There is also a partition record for each useable partition on the device. We will use this later on as well.
The basic outline of the file system is shown in **Figure 1**. There are a number of distinct areas on a mass storage device beside just the directory and data area. The partition information begins at offset $1BE_H$ of the first physical sector on the drive. After this sector, there are a number of reserved sectors, followed by the actual file allocation table. The FAT operates on groups of sectors called clusters. The FAT contains information about the physical clusters on the drive such as whether they are available to be written to and where the next cluster in the file is located. There are actually two copies of the FAT per drive for redundancy purposes. The root directory is 512 entries long in a FAT16 system with each entry occupying 32 bytes. Thus the total root directory takes up 32 sectors. After that is the data area containing files and subdirectories. The start of the data area is said to be at cluster number 2, although in reality it is actually more than 512 sectors from the beginning of the drive. **Table 1** shows the volume information for one particular USB flash device: The beginning of the actual data is 235_H sectors from the beginning of the drive. The area before this con-

Table 1. Volume information		
Volume Information	**Data Size**	**Value for current drive**
Fat Size	BYTE	FA_H
Total Sectors	DWORD	$3E7E0_H$
Total Data Sectors	DWORD	$3E5c8_H$
Number of Clusters	WORD	$F972_H$
Root Dir Sectors	BYTE	20_H
First Data Sector	WORD	235_H
Bytes Per Sector	WORD	200_H
Sectors Per Cluster	BYTE	004_H
First FAT Sector	BYTE	21_H
First Dir Sector	WORD	215_H

tains the partition record, boot record, two copies of the FAT, and the root directory. This information will be used later on to locate files on the drive.

SCSI

Many mass storage devices use SCSI commands for communication between the host and the device. We will be using three SCSI commands to perform our file operations. The commands are Inquiry, Read Capacity and Read(10). The format of the three SCSI commands to be used in this design is shown in **Table 2**.
Each command takes an Op Code and a Logical Unit Number (LUN). Some commands have an LBA field. An LBA is a Logical Block Address. A Logical Block is the smallest addressable section of a mass storage device, often referred to as a sector. Later on, we will see how to use an LBA to access file data. The Read (10) command also has a field for transfer length. Transfer length generally refers to the number of sectors to be sent or received. For the purpose of simplicity, we will work with only one Logical Unit. Thus, the LUN field will always be '0'.

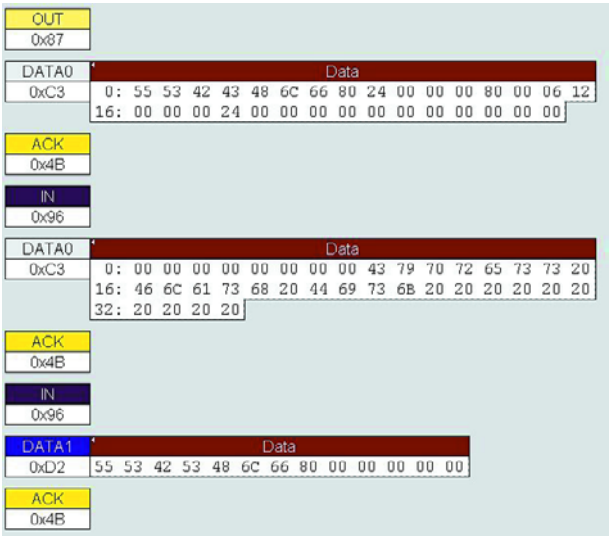


Figure 2. UFI transaction using CBW and CSW.

UFI

Removable media mass storage devices containing a USB interface use a common interface known as the UFI interface. The SCSI commands are embedded in the UFI packets. A UFI transfer is carried out in three phases: *command*, *data* and *status*. The command phase employs the *Command Block Wrapper* (CBW) and the status phase, a *Command Status Wrapper* (CSW).
The CBW is a 31-byte packet with the first 15 bytes containing four bytes of signature and four bytes of tag, three bytes of commands and flags and 4 bytes for the length of the actual data to be read or written. The final 16 bytes comprise the command block, which is the actual SCSI command. The CBW signature is used to identify the device and is hard-coded to 55 53 42 43 by the UFI specification. Interestingly, the signature in ASCII is 'USBC' for a USB Command. The tag can be any 4 byte value. The CSW will use the tag value received during the CBW phase to report the status of the CBW.
The CSW is a 13-byte packet and is similar to the CBW in that it contains four bytes of signature hard-coded to 55 53 42 53 which, in ASCII is 'USBS' for USB Status. There is also a 4-byte tag that will be matched against the tag of the CBW. The CSW also contains four bytes for the residue, which is the expected transfer length minus the actual transferred data length. There is one additional byte used to report transfer status. An example of a USB transfer using a CBW and a CSW is shown in **Figure 2**.
The first transaction is the CBW. The signature is contained in bytes 0 through 3 and the tag is contained in bytes 4 through 7. The tag can be any random 32-bit number. The CSW will use the tag value to Acknowledge completion of the command specified in the CBW. The '24' in byte 8 is a hexadecimal value representing 36 decimal and is the actual length of the transferred data in the second phase.
The second transaction is the data stage and includes the 36 bytes of data read back from the USB mass storage device. In this case, the data is in response to an Inquiry command (12_H) that can be seen in byte 15 of the first transaction. Converting the 36 bytes of data beginning at the ninth byte to ASCII will reveal the string 'Cypress Flash Disk'.
The third transaction is the CSW. Note the match between the tags of the CBW and the CSW. There is no remaining data to be transferred as shown in bytes 8 through 11. Byte 12 contains the status of the transaction. For this field, a '0' denotes success.

The Cypress EZ-Host controller

The Cypress EZ-Host is a 16-bit RISC based microcontroller, which contains two Serial Interface Engines (SIEs).

Each SIE controls two USB host ports or one USB peripheral port. The EZ-Host also contains a number of peripherals such as RS-232, SPI, HPI and HSS. There are 16 K bytes of internal memory space and provisions for an external memory interface.
The EZ-Host includes a firmware framework that takes care of most of the USB Host details. With full support for various types of USB transfers, reading and writing to USB mass storage devices is as simple as filling in some information and letting the frameworks handle the rest.
The EZ-Host framework contains all of the necessary firmware to implement USB Host functionality. Included in this are task scheduling, device enumeration, bandwidth allocation and power management.
Applications, on the other hand, are the firmware to control specific USB devices and communicate their data to an end application.
At the heart of the frameworks is the TD processor. The TD processor operates on a data structure called a Task Descriptor (TD) using its information to communicate with the USB hardware, specifically the Serial Interface Engine (SIE). It is important to note that each of the SIEs controls two ports and that there is a TD processor for each SIE.
The framework code incorporates a simple round-robin scheduler. The scheduler handles a number of tasks such as checking for device insertion and removal, checking the UART for incoming messages and executing the 'run' function of every installed device driver.
Basic USB transfers are used to move data in either direction (Host to Peripheral and Peripheral to Host). The host initiates the transfers. To move data to a peripheral, the host uses the function usb_send_bulk_out. To get data from a peripheral, the host uses usb_recv_bulk_in. Each of these functions requires some additional data – a pointer to the device structure, the USB endpoint to be used, the length of transfer, and a pointer to the send or receive buffer. Once these details are specified, the framework takes care of the rest.

Framework Flow

The framework code will execute as follows. On power on reset, the EZ-Host microprocessor will initialize all registers and counters as well as all device structures. It will then enter a loop:

– Check the host USB ports for any changes in status (devices inserted or removed).
– Check the TD processor and get status of all TDs running on the two SIEs.
– Go through the list of active device drivers and execute the run function for each.

Checking the host USB port for status changes requires inspection of a change variable. The port change interrupt handler will set this variable if a change occurs. If the port shows a change, enumeration code will be executed to service this change.
If a device is found, frameworks code will attempt to match the device to a registered driver. Driver matching can be accomplished in a number of ways. If there is only one specific device to be supported, the device's VID and PID can be added to a 'Targeted Peripheral List'. The device's VID and PID are then checked against this list. A more common method of device/driver matching is via device class and interface class. In the case of mass storage devices, both the device class and interface class have a value of '8' and their respective subclasses have a value of '6.' So, when the driver is created, these values will be entered into the appropriate fields (see **Figure 3**).

Table 2. SCSI commands

SCSI command	Inquiry	Read Capacity	Read(10)
Byte #0	12_H	25_H	28_H
Byte #1	LUN	LUN	LUN
Byte #2	Page Code	LBA (MSB)	LBA (MSB)
Byte #3	Reserved	LBA	LBA
Byte #4	Allocation Length	LBA	LBA
Byte #5	Reserved	LBA (LSB)	LBA (LSB)
Byte #6	Reserved	Reserved	Reserved
Byte #7	Reserved	Reserved	Transfer Length (MSB)
Byte #8	Reserved	Reserved	Transfer Length (LSB)
Byte #9	Reserved	Reserved	Reserved
Byte #10	Reserved	Reserved	Reserved
Byte #11	Reserved	Reserved	Reserved

Building an application

In this section we'll show how to build a simple embedded USB design that controls a USB Flash drive. First, we will need to create a driver for the mass storage device. We can call the driver the mass_storage_driver or some name indicative of the capabilities of the device. The most important fields in this data structure are the class and subclass fields. The framework will use these values to match this driver up with any Flash device with the same class and subclass values. One other possibility here is to enter a specific Vendor ID and Product ID in the driver template. In this case, only the device containing

```
CLASS_DRIVER    mass_storage_driver =
{
    0x08,           /* uint8   class;       */
    0x06,           /* uint8   subclass;    */
    0x08,           /* uint8   Mass storage if_class;    */
    0x06,           /* uint8   Mass storage if_subclass; */
    0x00,           /* uint8   Boot protocol;    */
    0x04B4,         /* uint16  vendor_ID;   */
    0xDE03,         /* uint16  product_ID;  */
    msdrvr_start,   /* uint16 (*start)( USB_DEVICE *dev ); */
    msdrvr_stop,    /* uint16 (*stop)(void);  */
    msdrvr_run      /* void (*run)(void);  */
};
```

Figure 3: Device Driver Template.

```
/*****************************************************************************/
/*                                                                           */
/*  ReadFile                                                                 */
/*                                                                           */
/*****************************************************************************/
uint32  ReadFile(USB_DEVICE *dev, uint32 Location)
{
    uint32  Sector, NextCluster = 0;
    uint16  Size, Result;
    uint8   Index, Done = 0;

    for(Index = 0;(Index < Volume->sectors_per_cluster) && !Done;Index++)
    {
        /*  we have to run through this n times, n being the number of sectors per cluster.  */
        /*  since we are getting passed in the cluster number but are reading by sector.     */
        /*  this function also needs to calculate the absolute location on the drive based   */
        /*  on the start_sector returned by FindFile note that Volume->first_data_sector     */
        /*  points to "sector #2" on the drive.  Use this when doing the sector math shown   */
        /*  on the next line                                                                 */
        Sector = Volume->first_data_sector + Volume->root_dir_sectors +
                 (Location-2)*Volume->sectors_per_cluster + Index;

        Result = SendCBW(dev, 0x200, 0x0a, 0x28, WordSwap(Sector), 0x100);

        Delay_mS(1);
        Size = FWX_SECTOR_BUFFER_SIZE;     /* Data */
        Result = usb_recv_bulk_in( dev, MSCPERPH_BULK_IN_PIPE, XferBuffer, &Size );

        BytesRead += 0x200;

        if(BytesRead >= FileSize)
            Done = 1;

        /* send the file data to the external process */
        PutData(XferBuffer, Size);

        Size = 13;      /* CSW */
        Result = usb_recv_bulk_in( dev, MSCPERPH_BULK_IN_PIPE, XferBuffer, &Size );
    }
    NextCluster = ReadFAT(dev, Location);

    return(NextCluster);
}
```

Figure 4. ReadFilr Code.

those Vendor and Product IDs will be started when plugged in.
Once the driver is created, the next steps are to add the start, stop and run functions plus the class codes for the device so the driver can be found. A driver structure is shown in **Figure 4**.
The name of the driver function has to be added to the file drvrlist.h:

#define FWX_DRIVER_LIST { &mass_storage_driver }

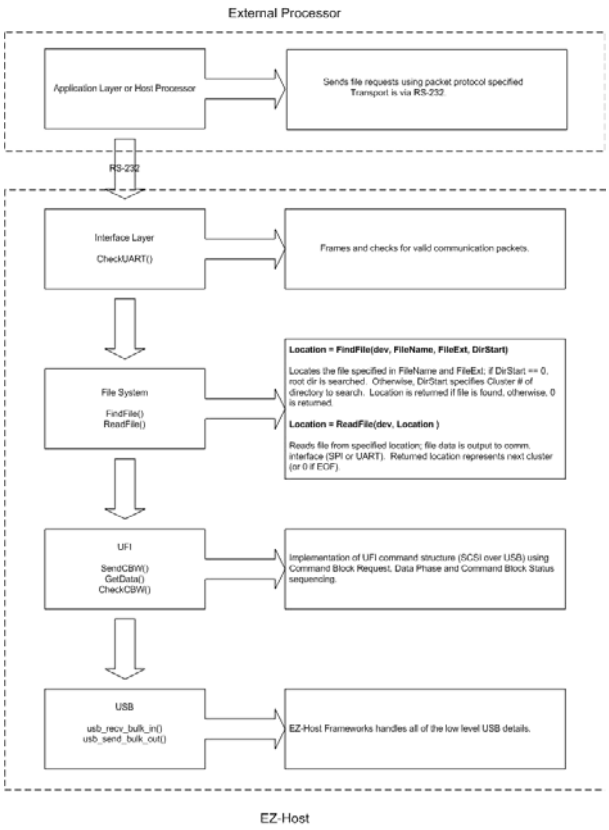


Figure 5: Simple USB Host Stack.

The mass_storage_driver's 'start' function will use the CBW/CSW protocol to interrogate the device to determine its characteristics. Code inside of these functions will get the data and pass it on to the application layer of the design. This will enable the higher-level layers of the application to handle the specifics of file management. The stop function will clean up any pending operations or transfers and notify the application layer that a device has been removed. The run function, finally, will be used to check for any incoming commands from the application code.

Using the frameworks to read a USB Flash drive

Now it is time to create some application code that actually does something. At this point, we have created a driver and have a means to match it up to a device by class and subclass matching. Next up is to add code to the driver for its start and run functions.

When the driver starts, we want it to go out and interrogate the flash device to gather information about its file system. Then, the run function will be used to periodically check as to whether the user wants to read any file data. The driver's start function will gather data about the file system in the following way:

– Perform a SCSI Inquiry command to get information on the attached device.
– Perform a SCSI Read Capacity command to get information about the size of the attached device.
– Perform a SCSI Read(10) with the LBA set to 0 to get data from the first physical sector on the flash device.
– Type cast the 16 bytes beginning at offset $1BE_H$ of the first sector to a partition record entry.
– Using the partition record information, locate the 'Start LBA' — the number of the cluster containing the boot record.
– Perform a SCSI Read(10) with the LBA value set to the Start LBA to read the sector containing the boot record. Use the information in the boot record to fill in the volume information.

At this point, we have used three SCSI commands and our knowledge of the drive layout to obtain the information that was presented in Table 1. We can now use this information to read any file on the drive.

The directory entry for a file will contain its Start Cluster number. This value will be passed to the file reading function in the Location field. The ReadFile function will read each sector in the cluster, then check the FAT to see if there are additional clusters for the file. The code required to perform a file read is shown in **Figure 4**. The Sector value can be calculated using information contained in the Volume structure. Note that the file reading loop executes once for each sector in the cluster. When a cluster has been fully read, the FAT is checked to find the NextCluster value. Also note the use of CBWs and CSWs for communicating with the drive.

Application Programming Interface

Figure 5 shows the hierarchy of the EZ-Host firmware. An external processor or application can communicate with the interface layer via RS232. Application code can be written to provide an API between the interface layer and the FindFile and ReadFile functions.
One possible interface layer is shown in **Figure 6**. Here, the UART is checked for a one byte Op Code followed by a filename, file extension and directory start location. An Op Code of '1' causes the FindFile function to be executed. An Op Code of '2' signifies a file read operation.

The FileName and FileExt fields in the packet specify the file to be found or read. The additional packet field, DirStart allows for the location of files in either the root directory (DirStart = 0) or in a sub directory (DirStart = start cluster of the subdirectory).

As an example, suppose we wanted to read a file called *File1* in the root directory:
The first UART Packet sent to the interface layer would have the parameter FileName set to *File1*, a DirStart value of '0' and an Op Code of '2' for a file read. The EZ-Host firmware would return the contents of the file *File1*.

Now, suppose we wanted to read a file called *File2* in the subdirectory *SubDir1*.

The first UART Packet would have the **FileName** parameter set to ***SubDir1***, the Op Code set to '1' for file find and the DirStart value set to '0' The FindFile function will return the starting cluster of the *SubDir1* directory to the application. Next, another UART packet would be sent, this time with the FileName parameter set to *File2*, the Op Code set to '2' for file read and the DirStart value set to the starting cluster of *SubDir1* which was returned from the previous FindFile function. At this point, the EZ-Host will return the contents of *File2*.

Using this API and the above examples, a file from any directory on the drive can be read.

Conclusion

Using an embedded Host USB controller, removable mass storage can be added to any design. This feature allows for downloading of critical data or for firmware upgrades to the design via USB.

An embedded USB host may be implemented with just one additional IC and no third-party software. An embedded USB Host framework has taken care of all of the lower-level details for us. Cypress' EZ-Host USB host can support the full speed Bulk transfers required to communicate with mass storage devices. The present code size is approximately 13 kbytes. File system support can easily be added to support application specific devices.

(040238-1)

```
/*****************************************************************************/
/*  msdrvr_run                                                               */
/*                                                                           */
/*****************************************************************************/
static void msdrvr_run( USB_DEVICE *dev )
{
    uint8   Status, Done = 0;
    uint32  NextSector = 1, Location = 0;
    char    FileName[9], FileExt[4], DirStart[5];

    if(FatType)                                 /* make sure volume is mounted  */
    {
        Status = CheckUART(FileName, FileExt, DirStart);

        if(Status == 1)                         /* find a file in a sub directory */
        {
            /* get the cluster number from the UART */
            Location =  GetCluster(DirStart);

            /* find the file location   */
            Location = FindFile(dev, FileName, FileExt, Location);

            /* send the next cluster number via the UART */
            if(DirStart != 0)
                PutCluster(Location);
        }

        else if(Status == 2)                    /* read a file given a cluster num */
        {
            /* get the cluster number from the UART */
            Location =  GetCluster(DirStart);

            while(!Done)
            {
                /* get the file contents        */
                NextSector = ReadFile(dev, Location);

                /* location of next file read   */
                Location = NextSector;

                /* check for breakout condition */
                if(NextSector == 0)
                {
                    Done = 1;
                }
            }
        }
    }
}
```

Figure 6: Example Interface Layer.

Web pointers

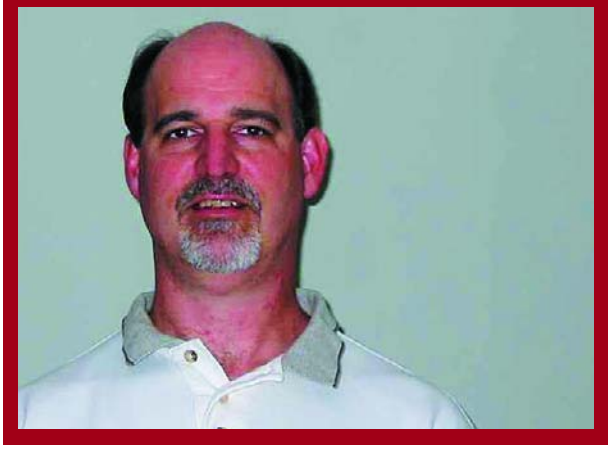
FAT file system:
www.microsoft.com/hwdev/download/hardware/fatgen103.pdf

UFI Command Specification:
www.usb.org/developers/devclass_docs/usbmass-ufi10.pdf

USB Mass Storage Class Bulk Only Transport Specification:
www.usb.org/developers/devclass_docs/usbmassbulk_10.pdf

Mark Schultz

Is a Senior Customer Design Center Engineer with Cypress Semiconductor. Mark holds a BSc in Information Technology from New Jersey College. Before arriving at Cypress, he worked on CCD camera bus interface boards at Princeton Instruments. Mark is a member of IEEE and has over 20 years experience in the electronics industry.



BOGUS ELECTRONIC PARTS

Beware of fakes!

Harry Baggen

http://sound.westhost.com/counterfeit.htm

The forgery of expensive brand-name products is an easy way to make some money quickly. Lots of products, anything from T-shirts to watches that strongly resemble the original are offered for sale, particularly in Asia, for ridiculously low prices. In recent years, even electronic components are being copied. From the outside they are indistinguishable from the original, but on the inside they are pure fake. This can have nasty consequences when you solder one in your circuit!

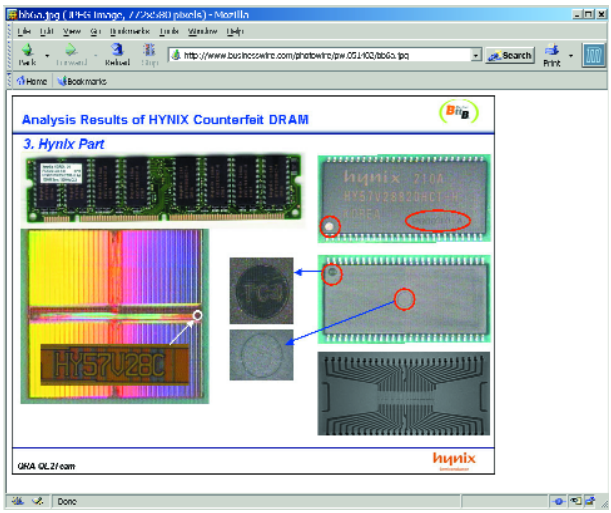
Forgery of brand-name products has been a phenomenon for a long time. Everyone will have heard the holiday stories where you can buy a 'real' Rolex on the beach for just 5 pounds! It is too good to be true…
But it is not only with clothing, handbags and watches that shady factories and slippery individuals try to make easy money at the expense of someone else's reputable name. These days, counterfeiters go much further. Everything that can earn money is a potential target for the forgery industry.

A very serious problem in the area of copied parts is fake parts for aircraft, so-called bogus parts. Aircraft parts have to comply with very stringent quality demands and are quite expensive as a consequence. A very attractive market for forgers, who will refurbish old parts or will attach other brand names to cheap parts in order to sell them on for a lot of money. But such fake parts can cause serious problems if they are used in or on an aeroplane. Experts have attributed several accidents in recent years to such 'bogus parts'.
The counterfeit parts turn up in all sorts of areas: clothing, watches, bank notes, pharmaceuticals, foods, toys, sunglasses, automotive parts, computer software and electronic components.

This problem is not just something that started in recent years, but has existed at least since the seventies. In 1981 a scandal was uncovered where millions of fake contraceptive pills were distributed, with all the consequences resulting from that. Now you can only smile about these things. Much more serious are fake aorta pumps for open-heart surgery, your life is then really hanging by a thread!

Even the space organisation NASA can't avoid the problem. They started legal proceedings last year against a company for supplying fake connectors.
In the electronics area you may remember the issue with fake capacitors on PC motherboards, a few years ago [1]. Even motherboards from reputable brands were fitted with electrolytic capacitors of very poor quality (but with the label of a well known manufacturer) that gave up the ghost after a very short time by exploding or by leaking the corrosive internal fluid all over the PCB.
Recently there was a problem with fake batteries for GSM mobile phones, which reportedly had a risk of exploding. This did not only involve batteries from unknown brands, but also fake batteries on which the labelling would indicate that they came from the 'official' manufacturers.

Experts suspect that most of these counterfeit parts originate in Asia, or China in particular. The damage caused by this is estimated to be several billion pounds (just for the electronic components). Because of the strongly growing industry and the increase of chip manufacturing facilities in China this is likely to increase significantly in the future. A good overview of recent counterfeit products can be found on the website for Designchain Associates [2].
There are a number of organisations worldwide that occupy themselves with fighting the 'counterfeit terror',

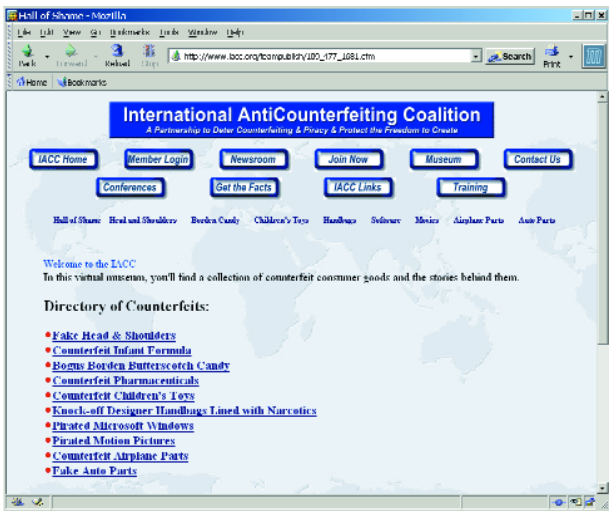


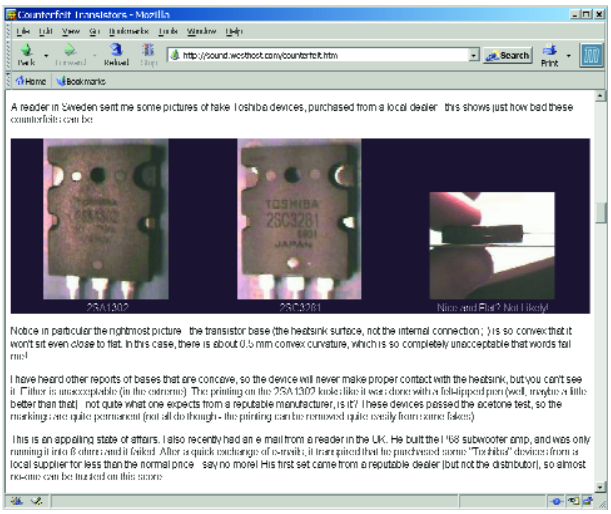
such as the IACC (International AntiCounterfeiting Coalition) [3].

Serious problem

With electronic components it is not so easy to check if a part meets the specifications of the manufacturer. You could perhaps run a few logic tests on a digital IC, but it is much harder with an analogue component. It is a big problem for manufacturers of equipment to determine in time (that is, before assembly) if all parts originate from a trusted supplier.

Most semiconductor manufacturers have recognised how serious this threat is for them and will warn their customers as much as is possible when counterfeit components have surfaced somewhere. These are often accompanied by detailed descriptions so users can determine the differences between authentic and fake for them-





selves. Various famous brands such as NEC, LG, Hynix, Agilent, Vishay, Altera, Atmel, Hitachi, Motorola and Toshiba have already had to deal with counterfeit copies of their products.

Quality

The quality of fake parts appears to vary enormously. There are (often passive) components that are so well made that it is very hard to tell that they are not from the original manufacturer. But in most cases the quality is significantly lower. Power transistors are a favourite subject for forgers. Here the packages are carefully copied, so that they look as much as possible as the original, but on the inside is a chip with much poorer characteristics. Usually the chip is also much smaller (i.e., cheaper!) than the original, so that the transistor will fail in a very short time. Several examples are already known from Toshiba and Motorola.

But it can be even worse. In one case, a company bought a batch of ICs (LT1040) that were desperately required, for a considerable amount of money because they were scarce on the semiconductor market. After inspection it was realised that there were no chips inside the packages!
Another phenomenon that has appeared in recent years, is offering microcontrollers with OTP memory. These can be programmed only once (One Time Programmable). Already programmed parts are sold as new and the new owner is then stuck with a batch of useless parts!

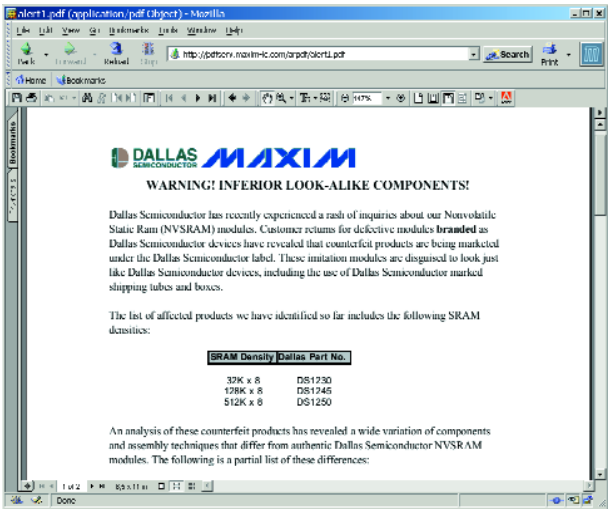
The moral of the story: Be alert!

There is no single way to provide 100% protection from fake parts. Semiconductor manufacturers recommend their customers to obtain their components only from the official distributors or trusted suppliers, but it has already transpired that even these channels can be corrupted with forgeries.

It is, in any case, a good start to buy from known suppliers and to check on the manufacturer's website if there are warnings for specific parts.
A number of companies have put together lists of fake components [9], with short descriptions of visible discrepancies with respect to the original parts. A glance at such a list is also recommended. In addition there is a special website for fake power semiconductors [10]. Everyone who builds or services audio amplifiers should certainly pay a visit there.

And finally the simplest advice: *caveat emptor*, be on your guard for deals that just look too good to be true. After all, you get nothing for nothing!
Readers with experience of fake parts are invited to contact the editor, because we are keen to find out how big this problem is in Europe. We can then also warn our other readers!

(045060-1)



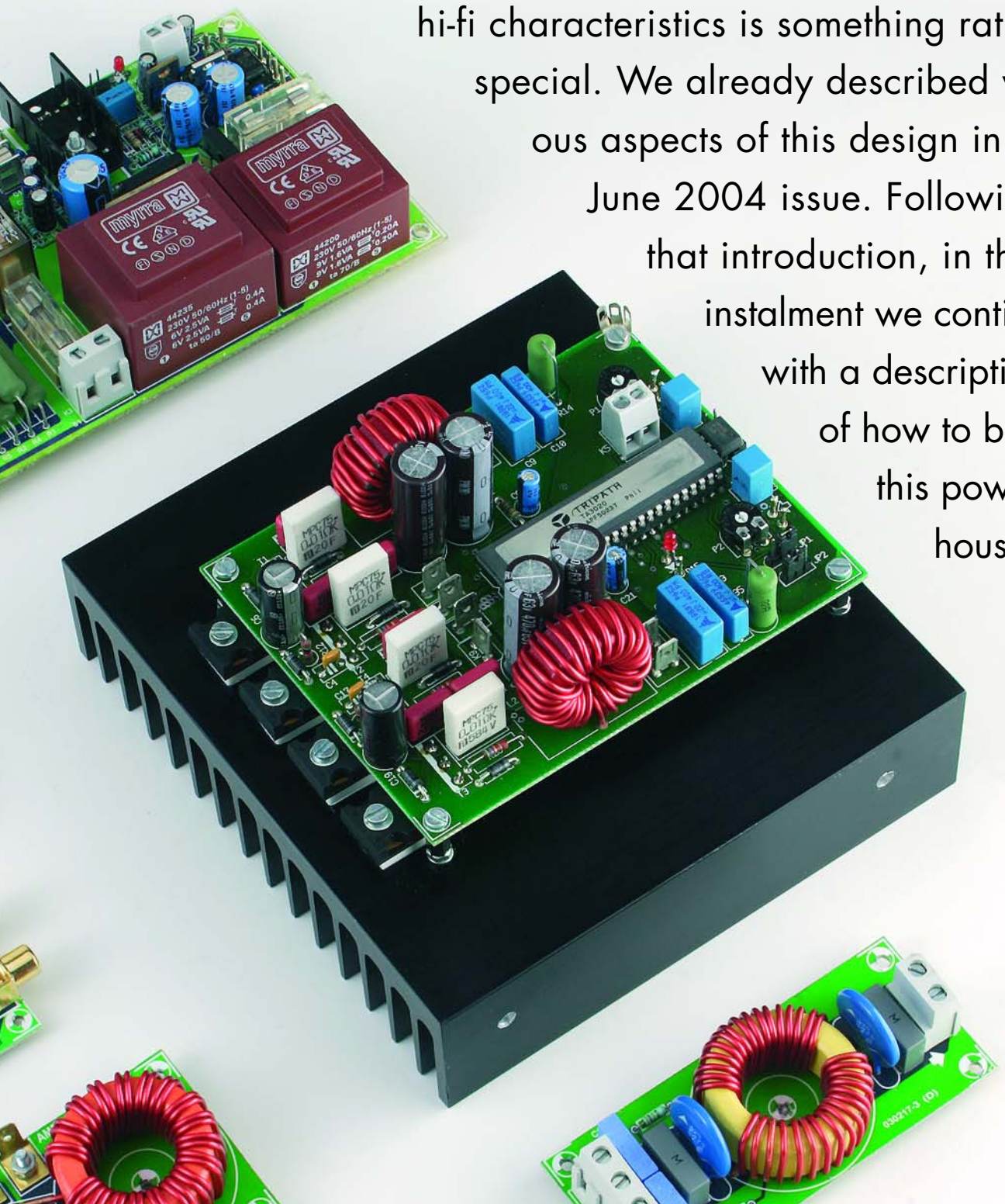
Internet addresses

[1] Leaking capacitors muck up motherboards:
www.spectrum.ieee.org/WEBONLY/resource/feb03/ncap.html#f1
[2] Designchain Associates:
www.designchainassociates.com/counterfeit.html
[3] IACC: www.iacc.org
[4] Maxim: http://pdfserv.maxim-ic.com/arpdf/alert1.pdf
[4] Kamaka: Counterfeit parts listing:
www.kamaka.de/deutsch/service/counterfeit-parts-listing.htm
[5] ESP: Counterfeit semiconductors:
http://sound.westhost.com/counterfeit.htm

ClariTy 2x300W Class-T

Ton Giesberts

A pulse-width-modulated output stage with hi-fi characteristics is something rather special. We already described various aspects of this design in the June 2004 issue. Following that introduction, in this instalment we continue with a description of how to build this power-house.



Amplifier

Part 2: building the amplifier board

The pulse-width modulated amplifier we're talking about here is based on the Tripath TA3020 driver IC. The term 'pulse-width modulated' may call up negative associations, but they are fully out of place here. The specifications are outstanding, and the amplifier is certainly as good as quite a few models in the upper end of the commercial range.

To make construction as easy as possible for *Elektor Electronics* readers, we can supply the printed circuit board for this project with all of the SMD components already nicely soldered in place, all at a very attractive price. And don't forget that a single circuit board houses a complete 2 × 300-W stereo amplifier.

Special components

Before you start enthusiastically buying components, we'd like to emphasise that you can't simply conjure up a dual 300-watt amplifier from of a box of standard parts. When this amount of power is involved, even the power supply must meet special requirements. On top of that, we're dealing with a switch-mode amplifier here. That makes the circuit board layout and the quality of the components especially important. To avoid potential problems, you need to know the requirements imposed on the various components before you start building the circuit board for this final amp.

Most of the components can be obtained from Farnell as stock items, and several merchants including C-I Electronics and Geist Electronics will be offering complete parts kits for this amplifier. It's essential for some of the components to be SMD types, due to the characteristics of SMDs and the short signal paths that can be obtained with them. However, this doesn't mean that soldering has to be a problem, since all of the SMDs are already fitted to the board. The ferrite cores for the inductors are also included with the printed circuit board (two per board). Winding the inductors with 1.5-mm enamelled copper does require a pair of sturdy fingers, but we have more to say about that later on.

Decoupling

The issue that requires the most attention with this amplifier is the interference that can be generated by quickly switching large currents. The circuit board has therefore been designed such that tracks that carry large currents have the least possible amount of coupling with the rest of the circuit. In addition, the supply voltages are always decoupled locally, in order to keep the loops in subcircuits conducting large currents as small as possible. In particular, this decoupling is provided by C5, C18, C32, C33, C36 and C37 for the output transistors. We selected 250-V MKT types for these decoupling capacitors, since they can better withstand extremely high switching currents.

Capacitors C6 and C19 also deserve special attention. For these two electrolytics, it is extremely important to have the lowest possible self-inductance and effective series resistance, as well as a good thermal rating.

The snubber networks (C4/R12 and C17/R33) help remove HF overshoots. To save space, R12 and R33 are mounted vertically. When fitting them, keep the loop as small as possible in order to keep their parasitic self-inductance as low as possible. As 1-W resistors are used here, you must allow for a somewhat larger diameter than usual (see **Figure 1**). For the capacitors (C4 and C17), we have selected 200-V ceramic types. This is because the maximum voltage across these capacitors can be nearly the full supply voltage (approximately 110 V between the positive and negative rails), or even more with any overshoots that may be present.

Suppressing inductive spikes

Due to the physical dimensions of the components, parasitic self-inductance and overshoots will always be present. The consequences of this, particularly the inductive spikes (back-emf) from the inductors in the output filters, can be partially suppressed by using Schottky diodes and ultra-fast-recovery diodes as clamping diodes. This job is performed by D3, D4, D6, D7, D10, D11, D13 and D14. Diodes in DO-15 packages (MUR120) are fitted to the circuit board for D3, D4, D10 and D11. Diodes D6, D7, D13 and D14, which are SMD components in DO-214AA packages (MURS120T3), are fitted on the solder side underneath the IC (**Figure 2**). Both types of diodes are rated at 1 A / 200 V, with a recovery time of only 25 ns.

MOSFET drive circuits

In the printed circuit board layout, special attention has also been given to possible loops in the paths between the driver outputs of the IC and the gates of the MOSFETs. These loops must be kept as small as possible. The loop in the drive circuit for the upper MOSFET of each channel consists of the HO driver, the gate resistor, the gate-source capacitance and the driver return connection (HO1COM or HO2COM). For the lower MOSFET, the loop consists of the LO driver, the gate resistor, the gate-source capacitance and the driver ground connection (LO1COM or LO2COM).

The overshoots arising from switching the MOSFETs are limited by the resistors incorporated in the gate circuits. This unavoidably leads to a compromise between the delays for switching

Components list

Pre-fitted SMD parts not listed. If necessary the list of pre-fitted parts may be downloaded from our website. Suggested suppliers are mentioned with unusual components only. These suppliers are not exclusive.

Resistors:

R6,R11,R27,R32 = 0Ω01, lead pitch 9mm, MPC75-E01 (H.O.D[1] , Bürklin[2])
R7,R9,R28,R30 = 5Ω6/1 W lead pitch 15mm (max.), PR01 BCComponents (Farnell # 337-584, 10+)
R12,R33 = 15Ω 1W, PR01, BCComponents (Farnell # 337-638, 10+)
R13,R34 = 240Ω
R14,R35 = 22Ω 5W (vertical))
P1,P2 = 10kΩ preset

Capacitors:

C1,C14 = 3µF3 50V, MKT, lead pitch 5mm or 7.5 mm
C4,C17 = 220pF 200 V, COG, lead pitch 5 mm, dipped radial multilayer ceramic, Multicomp (Farnell # 747-075, 1+)
C5,C18,C32,C33,C36,C37 = 100nF 250V, lead pitch 7.5mm or 10mm, w x l = 6 x 13 mm (max.), Wima MKS4 (Farnell # 148-888, 1+)
C6,C19 = 47µF 160V radial, lead pitch 5mm, diameter 10mm (max.), 105ºC, Panasonic EEUED2C470 (Farnell # 83-6400, 1+)
C8,C21,C38 = 47µF 25V radial
C9,C22 = 220nF 400V MKP, lead pitch 15mm, w x l = 8.5 x 18 mm (max.), Epcos B32652-A4224-J (Farnell # 400-3755, 1+)
C10,C23 = 100nF 400V MKP, lead pitch 15mm, w x l = 7 x 18 mm (max.), Epcos B32652-A4104-J (Farnell # 400-3731, 1+)
C30,C31,C34,C35 = 470µF 63V radial,

off one MOSFET and switching on the other one. For this amplifier, limiting resistors with a value of 5.6 Ω are recommended. This also allows part of the power that would otherwise be dissipated in the driver transistors to be dissipated in the resistors.

Another compromise is naturally the maximum power rating of the amplifier and the resulting MOSFET selection. Unfortunately, maximum drain current goes hand in hand with high gate-source capacitance (C_{iss} = 3800 pF max.). The parasitic self-inductance of the gate is also a factor; the lower it is, the faster the gate charge can be built up or removed.

To accelerate MOSFET switch-off, diodes are placed in parallel with the gate resistors. These are also ultra-fast-recovery diodes in 'normal' packages (MUR120). Thanks to their relatively large dimensions, they can easily bridge several broad tracks on the printed circuit board. A disadvantage of using these diodes is that the power dissipation in the drivers increases.

Due to their power dissipation, the gate resistors are 1-watt types. We have selected the very compact PR-01 series from BCComponents. Due to the construction of these metallic-film resistors (helical groove), they unavoid-

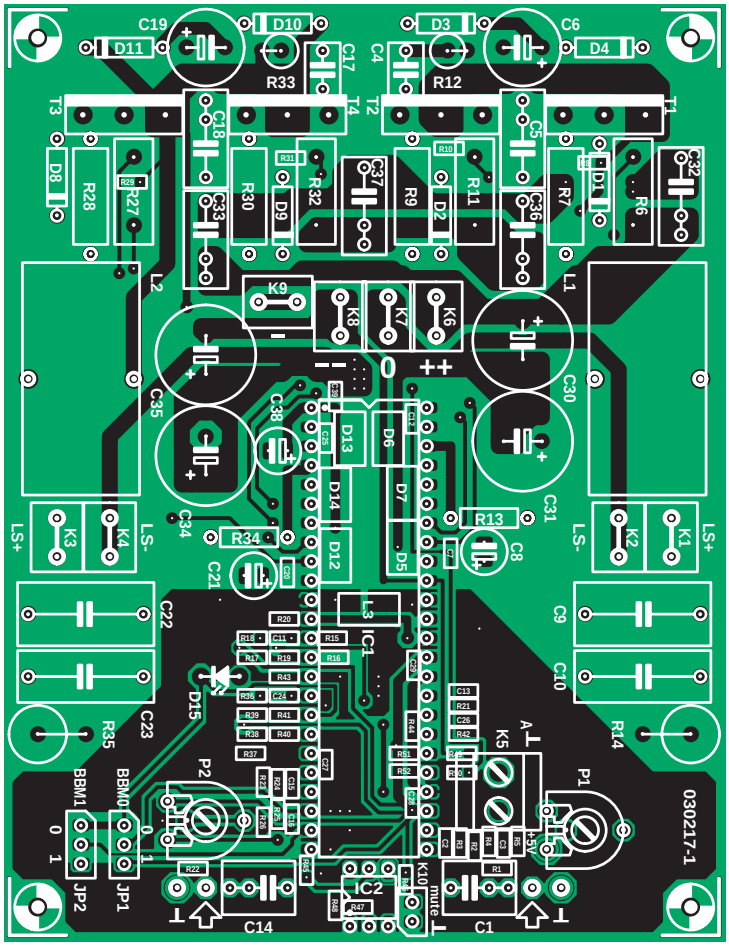


Figure 1. Component layout (top side of the amplifier board).

lead pitch 5mm, diameter 13 mm (max.), 105ºC, Nichicon UPM1J471MHH (Farnell # 415-3030, 5+)

Inductors:
L1,L2 = 11µH3, 29 turns 1.5 mm ECW (SWG 16) on T106-2 core (Micrometals) (core supplied with pre-fitted PCB)

Semiconductors:
D1-D4,D8-D11 = MUR120 1 A/200 V ultra fast, ON Semiconductor (Farnell # 930-994, 1+)
D15 = LED, red, high efficiency
T1-T4 = STW38NB20, TO-247 case, 200 V/38 A, ST (Farnell # 323-9408, 1+)
IC1 = TA3020, Tripath[3]
IC2 = CNY17-2

Miscellaneous:
JP1,JP2 = 3-way pinheader with jumper
K1-K4,K6-K9 = spade terminal, vertical, PCB mount
K5 = 2-way PCB terminal block, lead pitch 5 mm
K10 = 2-way pinheader
48-pin IC socket, DIP socket with turned pins 0.6" (15.24 mm) row spacing (Farnell # 416-8653, 1+)
4 ceramic washers AOS220SL, Fischer, 14 x 18 mm, 4.5 mm thick (Huijzer-Avera[4])
Heatsink 0.6 K/W, 160 x 150 mm, Marston 938SP01500A200 (Farnell # 526-794, 1+)
PCB, order code **030217-91**. Comes with all SMD parts pre-fitted and cores for L1, L2 supplied.

1. www.hod-electronics.nl
2. www.buerklin.de
3. sales@profusionplc.com
4. www.huijzer.com/

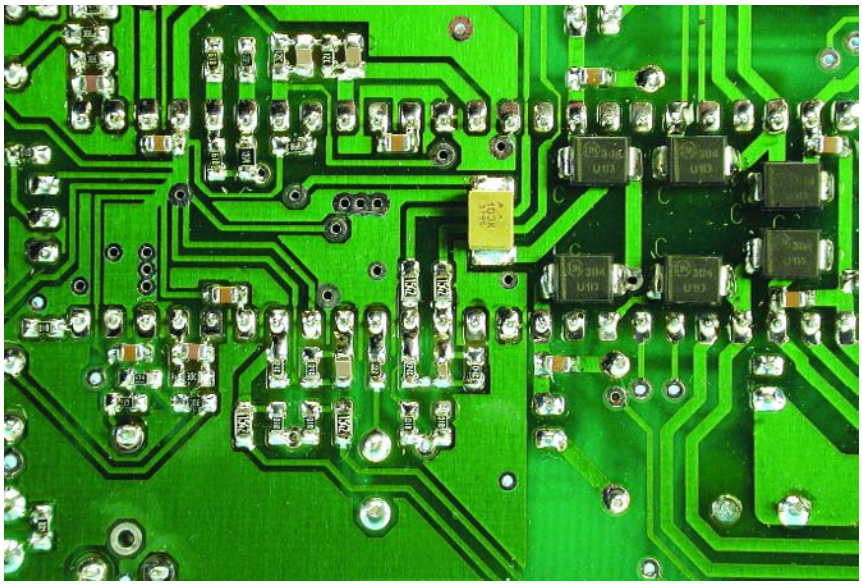
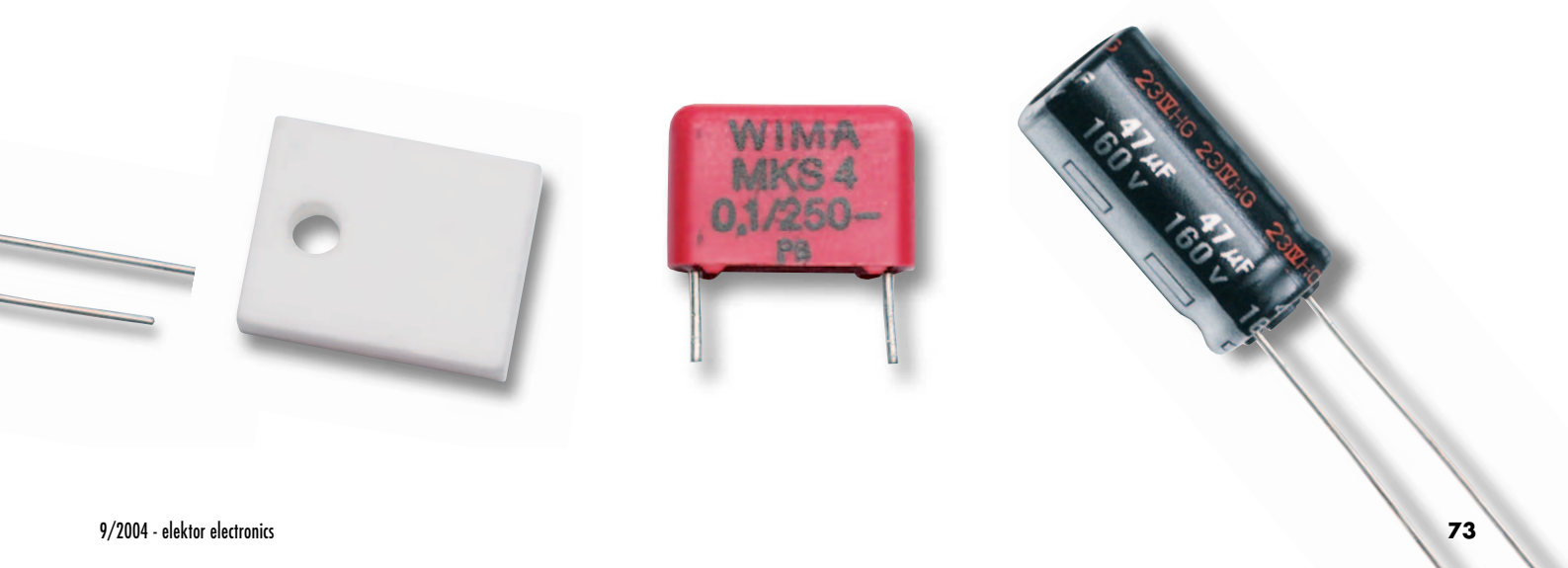


Figure 2. Bottom side of the amplifier board. The SMDs are pre-fitted.

ably have significant parasitic self-induction, but the impedance of this resistor type is still relatively constant up to 10 MHz. A good alternative would be carbon compound resistors, which have much lower self-inductance due to their construction. Sufficient room is provided on the circuit board for the latter type.

Power supply and ground

The main supply voltage is connected to the circuit board using flat (car-type) terminals. This allows very large currents to flow and makes it easy to connect the board to the power supply. Special electrolytic capacitors are fitted across the power supply terminals to decouple the worst RF current spikes. We have done our best to implement these connections as star points, but we had to spread them out a bit to keep the distance between the IC and the MOSFET leads reasonably short.
For decoupling capacitors C30, C31, C34 and C35, we selected a Nichicon family that combines a very good capacity/size ratio with low series resistance and low self-inductance. The optimum choice from this family is the 470-µF type with a working voltage of 63 V. In our case, this determines the maximum allowable supply voltage for the final amp.
From the star point, a track runs to the inductor (L3) that isolates the analogue and digital grounds of the IC. This inductor comes in an SMD package (1812A) and is located on the solder side under the IC (**Figure 2**). It is a member of the Epcos SIMID family and has a value of 10 µH, with a series resistance of less than 1 Ω and a current rating of more than 300 mA.
The ground connections for the loudspeaker outputs are also tapped off from the star point, so the currents from the loudspeakers are conducted back to the main power supply as directly as possible. This avoids inter-



Pre-fitted circuit board

Due to the special nature of this amplifier (particularly the high switching frequencies), the choice of components is especially important and the use of SMDs in various locations is unavoidable.

Most amplifier builders have little or no experience with soldering these miniature components, and for this reason we provide this circuit board with all of the SMDs already fitted. All you have to do is to fit the IC and the 'normal' components. In addition, two cores for the output inductors are provided with the board, since they are made from a rather special material.

The price of the circuit board and inductors is only **£34.50 / $55.70 / €49.** This is a two-channel design, so you only need one circuit board for a stereo amplifier.

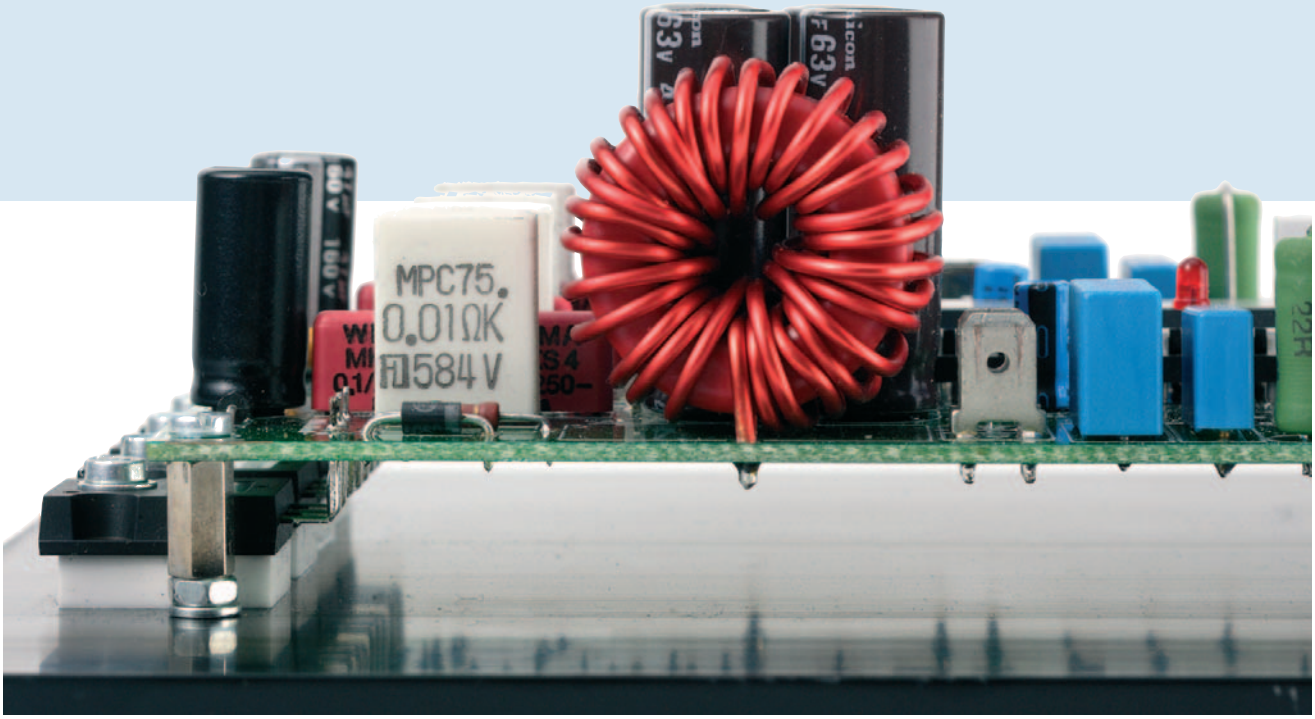


Figure 3. For the output inductors, several turns must be placed on top of other ones. The output transistors are fitted to the heat sink below the circuit board using special insulators.

ference to other parts of the amplifier. The ground planes on the circuit board are exclusively intended to provide protection against interference. They are only connected to the AGND pin for the analogue power supply (pin 28). This means that no other connections to the amplifier are tapped off from the ground planes.

A star point is also used for the connections between the analogue +5-V supply and the various components, including the trimpots, jumpers and resistor bridges for the modulator settings. This can be seen on the component side of the layout in **Figure 1**.

Construction

Despite all the attention given to the layout, the level of interference generated by the output stage is rather high. This means that additional measures must be taken to minimise the negative effects on the analogue portion. This is done by keeping the surface area needed to fit the passive components as small as possible. The only way to achieve this is to implement practically everything using SMD components, and to place them as close together as possible underneath and beside the IC on the solder side of the board (see **Figure 2**).

The only components that depart from this rule are the two trimpots for DC offset adjustment and the input capacitors. Any interference picked up by the trimpots is filtered out by C3 and C16. Interference picked up by C1 and C14 is filtered out by C2 and C15. Some of the SMDs must have a voltage rating greater than 50 V, due to their location in the circuit. This applies to R15, R16, R36, R37 and R51. SMD components are also used for R8, R10, R29, R31 and several decoupling capacitors, since this gives better functional results and takes up less space.

For readers who wish to take a soldering iron to the SMDs, despite the fact that the board is provided with the SMDs already fitted (for example, to change the input sensitivity or modify the board for a lower supply voltage), the solder lands are designed to allow either 0805 or 0603 packages to be used for all SMD capacitors and resistors. They are thus somewhat larger than the standard size for reflow or wave soldering. You can thus touch a soldering iron with a very fine tip on the solder

Kelvin connections

Kelvin connections (which are also called four-point measurements, depending on the application) are used at three locations in the amplifier to eliminate the effects of contact resistance and parasitic self-inductance.

A Kelvin connection is a connection to the terminals of a specific component for accurately measuring the voltage across the component without using extra taps. Here such connections are used for the voltages across the sense resistors for overcurrent detection, the feedback from the loudspeaker terminals and the input ground.

For current detection, it should be evident that four-point measurement is necessary, since the resistance of the sense resistor is only 10 mΩ. The overcurrent sense leads are tapped off directly from the resistor leads. The resistors are upright models in ceramic packages (MPC75 from Fukushima Futaba Electric Co. Ltd.), and they have practically zero self-inductance. If you cannot obtain these resistors, you will have to look for other low-inductance resistors with the same form. Resistors with axial leads are truly unsuitable, since fitting them vertically is by itself enough to produce too much self-inductance.

Each pair of signal leads is routed close together to the pins of the IC. The same is true of the feedback from the loudspeaker terminals (to the corresponding resistors). For the input ground, using a Kelvin connection means that all ground connections are individually routed to the common ground pin of the IC (the '0V' terminal for the analogue +5-V supply). This can be clearly seen on the solder side of the amplifier circuit board. Quite a few circuit board tracks join together here, with the result that this junction is necessarily somewhat broadened.

lands while soldering, which considerably simplifies soldering SMDs.
For the capacitors in the output filters, we have chosen 400-V polypropylene types that are especially suitable for applications with extreme pulse loading. Here we also use a compact version. These capacitors have a pitch of 'only' 15 mm, which results in a low value of parasitic self-inductance here as well. There is enough room here for somewhat wider components if you wish to use a different type (such as polyester or capacitors from a different manufacturer).

Coil winding

Winding the output inductors is not difficult, but you must pay careful attention to the winding method. With the selected wire diameter of 1.5 mm (16 SWG), the 29 turns will not fit on the selected core in a single layer. To keep the internal capacitance as small as possible, the coil is wound progressively in approximately seven sections. This means that after the first three turns are wound, the fourth turn is placed on top of the third turn, and the fifth turn is then wound directly on the core next to the third turn. It is followed by turns six and seven, with turn eight again being placed on top of turn seven, turn nine again next to turn seven, and so on (see **Figure 3**).

The relatively thick wire doesn't make the job any easier. Depending on how neatly and tightly you manage to wind the coil, you may well have to place a few more turns on top of other ones.

IC and output transistors

Most people will probably be reluctant to solder the IC to the circuit board, so we searched for a 48-pin IC socket with very high quality. We finally decided on a type having turned contacts with 30-micron gold plating and rated at 3 A. Here we can't be satisfied with anything less.
A very important detail in fitting the output transistors is the material for the electrical insulators. As these transistors have a metal cooling surface that is electrically connected to the drain (T2 and T4), the capacitance to the heat sink (which is connected to ground) would be too large if they were fitted using standard insulators made from mica, silicone rubber, silicone foam or even fancy stuff like Kapton. We tried this, and at maximum output power there were large parasitic currents that could not be suppressed.
There is only one good solution to this problem, which is to use ceramic (Al_2O_3) insulators that are several millimetres thick. The insulator we use here is the Fischer type AOS220SL, which is 4.5 mm thick and is actually intended to be used with a T0-220 package, instead of the larger TO-247 package. Despite being a bit too small, the insulator fully covers the metal cooling surface of the transistor. It also keeps the parasitic capacitance extremely small.
For the heat sink, we found a type with a surface large enough to mount the circuit board parallel to the surface. The selected type (from Marston) is 160 mm wide and 150 mm deep, and it even provides a bit of clearance at the edges. Eight holes tapped for 3-mm screws can be drilled in the base, which is 10 mm thick, for fastening the circuit board and the four output transistors. We recommend that you first centre the board on the heat sink and mark the four corner holes. Next, bend the leads of the output transistors exactly where they become thinner, slide them in place and mark the positions of the four fastening holes for the transistors.

Cylindrical standoffs (metal types with a threaded end) with a length of approximately 10 mm should be used to attach the circuit board to the heat sink. The threaded end will probably be too long to be fully screwed into the heat sink. This can be solved by first fitting a nut and lock washer on the threaded end. This causes the mounting height of the circuit board to be just right, and the leads of the output transistors will pass through the matching holes in the circuit board with length to spare.

Coming up

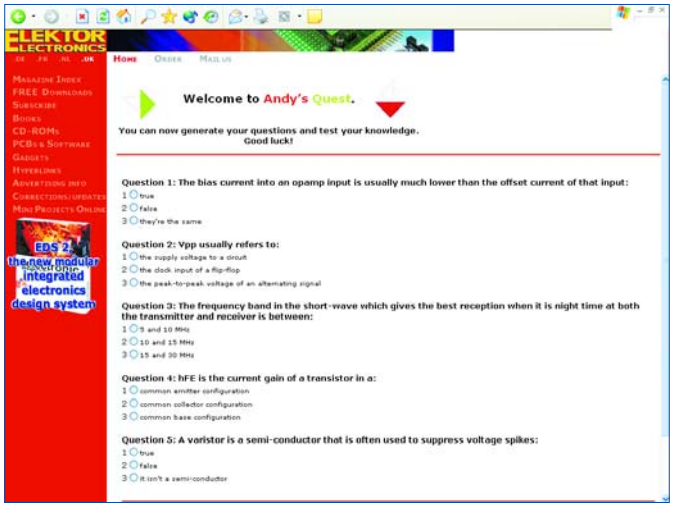
All that's left now is the power supply and the wiring diagram. We'll deal with them in next month's issue, when we'll also present some measured results for the fully assembled amplifier.
Items such as input and output filters and EMC problems will be handled in a separate article, which will also appear next month.

(030217-2)

Andy's Quest

A new quiz has been launched on our website www.elektor-electronics.co.uk in which you can test your knowledge of electronics theory. Participation is free and anonymous.

Each round of Andy's Quest draws a random selection of five multiple choice questions from a base of about 400. Have fun!



Railrouter project software

The author of the project, Mr. Ray King, kindly advised us that the 'Serial OCX' module he originally used in the Railrouter software was protected by copyrights owned by Everyday Practical Electronics (EPE) magazine. A new version of the compiled Railrouter software, eliminating the dependency on EPE's Serial OCX utility, is now available from the Free Downloads section of the Elektor Electronics website. We apologize to EPE for having distributed their copyrighted OCX module during the period 16 through 28 June 2004.

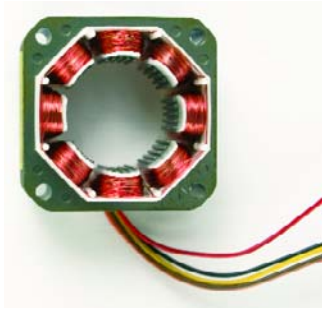
C for AVR Micros

A number of readers keep asking us about C compilers for Atmel AVR microcontrollers. We can recommend AVR-GCC for stability and the fact that it's free! See www.avrfreaks.net

---

Out of Step Dear Jan, I believe the photograph printed as the head of your article 'First Steps' in the May 2004 issue shows the internals of a synchronous three-phase motor, not a stepper motor. The letters printed near the three solder pads in the centre provide clues to the motor having been taken from a floppy disk drive.
D. Jansen, The Netherlands

Our apologies and we make amends by printing these photos of a real stepper motor.



Clarity Class-T Amp

Dear people at Elektor, congratulations on the awesome class-T amplifier (June 2004). But where is the chip available from in UK?
I managed to locate the MOSFETs at www.farnell.co.uk, but a search of the TA3020 revealed nothing. Many thanks for a cutting edge publication.
Gerard Galvin

Fishing for chips

Dear Editor, first let me congratulate you on the new look magazine; it is a big improvement. I imagine your main readers are professional electronics engineers who have ready access to obscure chips direct from the suppliers. As an amateur I find many of your projects of interest to me cannot be built because they depend on chips that not available in single quantities (even after a search on the Internet worldwide). In particular the high-end pre-amp and the versatile final amplifier of a few years ago. The same problem arises with your new Class-T amplifier project. These all depend on a single crucial, but unobtainable, chip. Would it be possible to supply these through your magazine or at least for the authors to specify a supplier?
Malcolm Read

Here is the contact information for Tripath's UK distributor: Profusion plc, Aviation Way, Southend-on-Sea, Essex SS2 6UN. Tel: +44 1702 543500, Fax: +44 1702 543700. sales@profusionplc.com

Two internationally operating suppliers of kits and components specifically for Elektor projects are:
Geist Electronic, www.geist-electronic.de
DIL Electronics, www.dil.nl

PA300 Amplifier

Hi again Jan, I'd like to construct the PA300 amplifier, but I'd like to know if this is possible.
I don't have the 625-VA transformer required, but instead I have two of 300-VA 40-0-40V transformers which I was going to use to build a stereo version of the PA300. Now I understand that I will not achieve 300 watts into 4 ohms per channel based on 300 VA.
What I'd like to know clearly is what the estimated power output will be for 8 & 4 ohms on 300 VA. I will use one transformer per channel.
I know there will be reduced power which will be enough for my needs but like to know what that is and for example will there be higher distortion?
Also I have 10,000 uF capacitors rated at 63 V, can I use 63-V capacitors in the power supply instead of 100 V as specified in the parts list? As my transformer delivers 40 Vac per rail this would be about 56 Vdc, so well below 63 V.
Can I still build this amplifier with this reduced power supply and is the setup of the quiescent current the same?
I've always wanted to build

this amp but haven't been too sure on a few things. I've heard many good things about it.
Ed Zammit (Melbourne)

Well Ed the lower transformer voltage will cause a supply voltage reduction of about 4 volts (at zero load). This will only decrease the maximum power output. The music power to be expected is 260 W into 4 ohms and 150 W into 8 ohms. Continuous power will be even less, due to the reduced power rating of the transformer and is hard to predict, because of unknown overload properties.

USB-to Analogue Converter Dear Jan, in the November 2003 issue, downloads for the USB Analogue Converter were stated to be available from the Elektor website. However, the HEX file used in the project (Elk.hex) was not one of the files included in the zipped file 020374-11 (hex code and Windows software). Also, it was not clear what files were required in the MPLAB project to compile and program the PIC16C765. If possible, could you provide the necessary information to enable me to finish this excellent project.
Kieran McAreavey

Several readers have asked basically the same question. The required HEX file (ELK35.HEX) is in the zip file under the directory 'fic_ass'. The source code provided is not guaranteed to be of direct use in MPLAB. It is not clear which assembler the author of the article used, so we are unable to help you with assembling the source code using MPLAB. This should be fairly easy however. Due to the fact that the HEX-file is included there is no direct need for you to assemble the source-code again, unless you want to modify the firmware to suit your own requirements.

High-End Preamplifier
Dear Editor, by now this should be a very popular question: what is the manufacturer's name (and P.N.) for the 16x2 LCD used in the High-end Preamp (April and May 2004)? I mean the one used by the author, Benjamin Hinrichs (the audiophile community owes him a lot!). I don't know if there's a standard interface and any equivalent part would fit....
Sergiu Ignat (Montreal)

Farnell (www.farnell.com) have a series of All Shore Industries, Inc. in stock. Look for order code 412-7316. The module Mr. Hinrichs used is of unknown origin.

Simple 12-to-230V Power Inverter Dear Editor, I am a teacher at a Polytechnic and would like to use the Simple 12-to-230V Power Inverter (February 2004) as a classroom project. Unfortunately I can not quite grasp the function of the 'R-C-Z diode' combination (D4, R14, C9, C10) in your circuit. Is this some kind of protective circuit? Also, what is the specification for the heat-sink as that information is missing from the article.
Oliver Sluka (Austria)

When T1 or T2 switches off, harmful voltage peaks may occur at the drain because an inductance (in fact, an non-ideal transformer) is seen between X2-X4. Diode D5 shunts these peaks into C9 and C10. These capacitors have to be discharged, however, to prevent them building up excessive voltage levels. As soon as the voltage on C9 exceeds the supply level plus the zener voltage (i.e., approx. 30 volts), the charge is shunted off preventing the capacitor voltage from rising further. D5 and D6 ensure that C9 and C10 do not affect the operation of te inverter under normal circumstances. Without these diodes, the capacitors could give rise to resonance. D5 and D6 only serve as damping devices when the voltage exceeds 30 V.
The heat sink specification is 10 K/W minimum.

Corrections & Updates

Build Your Own DRM receiver
March 2004, p.12, 030365-1
New software has been developed for this project that allows an USB/RS232 adapter to be used. This means the receiver can now be controlled from a laptop without a serial port. Free Download, see September 2004 items.

First Steps
May 2004, p. 30-34, 030203-1
In the circuit diagram the value of R17 and R18 is shown as 10 k, while 1 M appears in the parts list. The value is not critical but in practice 10 k will provide the best protection against noise at the FET inputs.

Personal Sound to Light Unit
June 2004, p. 58-60, 030019-1
The type code of IC1 should be LM358, not LM385 (text, parts list and schematic).

DDS RF Signal Generator
October 2003, p.14-22, 020299-1
Constructors are advised that a new version of the controller firmware is now available (ref. 020299-41). The update solves the problem of the software crashing when the 'B' key is pressed to switch between dBm and V. For a free update, return your controller chip with the Elektor ESS sticker on it to Readers Services, att. Mr. J. Visser, Software Service.

MailBox Terms

– Publication of reader's correspondence is at the discretion of the Editor.
– Viewpoints expressed by correspondents are not necessarily those of the Editor or Publisher.
– Correspondence may be translated or edited for length, clarity and style.
– When replying to Mailbox correspondence, please quote Issue number.
– Please send your MailBox correspondence to: editor@elektor-electronics.co.uk or Elektor Electronics, The Editor, P.O. Box 190, Tunbridge Wells TN5 7WY, England.

this amp but haven't been too sure on a few things. I've heard many good things about it.
Ed Zammit (Melbourne)

Well Ed the lower transformer voltage will cause a supply voltage reduction of about 4 volts (at zero load). This will only decrease the maximum power output. The music power to be expected is 260 W into 4 ohms and 150 W into 8 ohms. Continuous power will be even less, due to the reduced power rating of the transformer and is hard to predict, because of unknown overload properties.

USB-to Analogue Converter Dear Jan, in the November 2003 issue, downloads for the USB Analogue Converter were stated to be available from the Elektor website. However, the HEX file used in the project (Elk.hex) was not one of the files included in the zipped file 020374-11 (hex code and Windows software). Also, it was not clear what files were required in the MPLAB project to compile and program the PIC16C765. If possible, could you provide the necessary information to enable me to finish this excellent project.
Kieran McAreavey

Several readers have asked basically the same question. The required HEX file (ELK35.HEX) is in the zip file under the directory 'fic_ass'. The source code provided is not guaranteed to be of direct use in MPLAB. It is not clear which assembler the author of the article used, so we are unable to help you with assembling the source code using MPLAB. This should be fairly easy however. Due to the fact that the HEX-file is included there is no direct need for you to assemble the source-code again, unless you want to modify the firmware to suit your own requirements.

High-End Preamplifier
Dear Editor, by now this should be a very popular question: what is the manufacturer's name (and P.N.) for the 16x2 LCD used in the High-end Preamp (April and May 2004)? I mean the one used by the author, Benjamin Hinrichs (the audiophile community owes him a lot!). I don't know if there's a standard interface and any equivalent part would fit....
Sergiu Ignat (Montreal)

Farnell (www.farnell.com) have a series of All Shore Industries, Inc. in stock. Look for order code 412-7316. The module Mr. Hinrichs used is of unknown origin.

Simple 12-to-230V Power Inverter Dear Editor, I am a teacher at a Polytechnic and would like to use the Simple 12-to-230V Power Inverter (February 2004) as a classroom project. Unfortunately I can not quite grasp the function of the 'R-C-Z diode' combination (D4, R14, C9, C10) in your circuit. Is this some kind of protective circuit? Also, what is the specification for the heat-sink as that information is missing from the article.
Oliver Sluka (Austria)

When T1 or T2 switches off, harmful voltage peaks may occur at the drain because an inductance (in fact, an non-ideal transformer) is seen between X2-X4. Diode D5 shunts these peaks into C9 and C10. These capacitors have to be discharged, however, to prevent them building up excessive voltage levels. As soon as the voltage on C9 exceeds the supply level plus the zener voltage (i.e., approx. 30 volts), the charge is shunted off preventing the capacitor voltage from rising further. D5 and D6 ensure that C9 and C10 do not affect the operation of te inverter under normal circumstances. Without these diodes, the capacitors could give rise to resonance. D5 and D6 only serve as damping devices when the voltage exceeds 30 V.
The heat sink specification is 10 K/W minimum.

Corrections & Updates

Build Your Own DRM receiver
March 2004, p.12, 030365-1
New software has been developed for this project that allows an USB/RS232 adapter to be used. This means the receiver can now be controlled from a laptop without a serial port. Free Download, see September 2004 items.

First Steps
May 2004, p. 30-34, 030203-1
In the circuit diagram the value of R17 and R18 is shown as 10 k, while 1 M appears in the parts list. The value is not critical but in practice 10 k will provide the best protection against noise at the FET inputs.

Personal Sound to Light Unit
June 2004, p. 58-60, 030019-1
The type code of IC1 should be LM358, not LM385 (text, parts list and schematic).

DDS RF Signal Generator
October 2003, p.14-22, 020299-1
Constructors are advised that a new version of the controller firmware is now available (ref. 020299-41). The update solves the problem of the software crashing when the 'B' key is pressed to switch between dBm and V. For a free update, return your controller chip with the Elektor ESS sticker on it to Readers Services, att. Mr. J. Visser, Software Service.

MailBox Terms

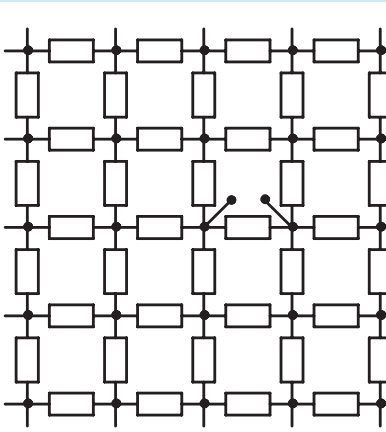
– Publication of reader's correspondence is at the discretion of the Editor.
– Viewpoints expressed by correspondents are not necessarily those of the Editor or Publisher.
– Correspondence may be translated or edited for length, clarity and style.
– When replying to Mailbox correspondence, please quote Issue number.
– Please send your MailBox correspondence to: editor@elektor-electronics.co.uk or Elektor Electronics, The Editor, P.O. Box 190, Tunbridge Wells TN5 7WY, England.

Quizz'

Professor Martin Ohsmann

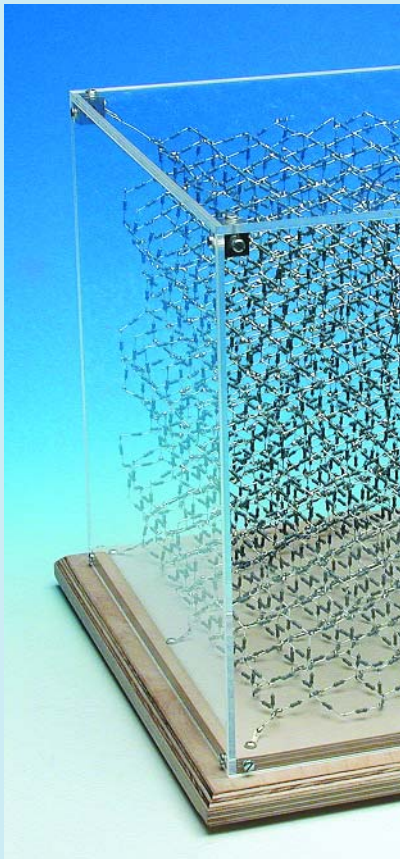
Martin Ohsmann was born in 1959 and as the photograph shows, got the hang of 'all things electric' (including hammers) at a very early age. Martin read electrical engineering at Technical University Aachen (RWTH), Germany, graduating in 1984. He became a staff member of the RWTH Mathematics Faculty (dissertation 1988), development engineer, scientific staff member of Philips Research Labs in Aachen. Since 1999 Martin is a Professor of Electrical Engineering and Information Technology at FH Aachen, lecturing in Operating Systems and Distributed Systems.
Martin Ohsmann has been reading the German version of Elektor since 1972. His first contribution to the magazine was an article on RDS decoding using a PC; this was published in 1989.

Many of you will have substantial amounts of identical resistors lying around, perhaps bought years ago as a 'super pack' or some other 'scoop' offer. To practice the noble art of soldering, dozens of these components may be soldered together to form a square grid array (**Figure 1**). Supposing you had an infinite number of resistors available, the array, too, would be infinite. This month we kick off our Quizz'away series with this problem:



A square array of infinite size is built from identical resistors with the value R = 1 kΩ. What is the resistance measured along a single resistor as shown in the illustration? How is the measured value explained?

If you feel like solving the problem the practical way, simply start soldering away a few dozen resistors and perform a simple measurement in order to get an approximate answer. Alternatively, why not push the limits of your electronics simulation program? Note however that we're not satisfied with just the value, but also require a well-substantiated explanation, comprising as little mathematics as possible, please!
It is also possible to have a go at three-dimensional arrays like cubes (**Figure 2**). Have fun!



Supplementary question

If you are not convinced that your explanation with the value measured in Figure 1 makes sense or is as simple as possible, you have a second chance to participate and win in this quiz. In the cube shown in Figure 2, the resistors form a three-dimensional array resembling the crystal structure of a diamond.

The supplementary question is:

How many resistors is the cube made of?

AWAY

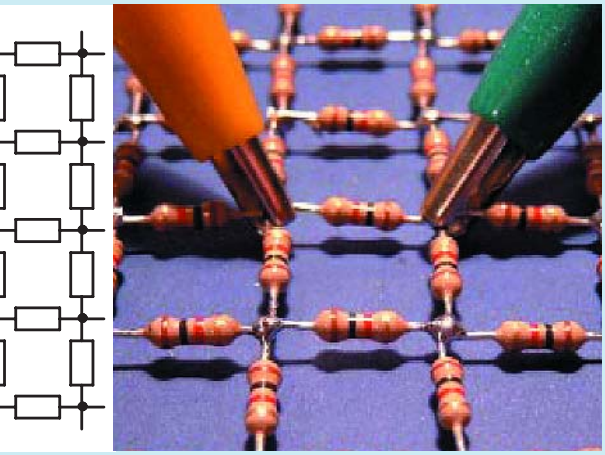
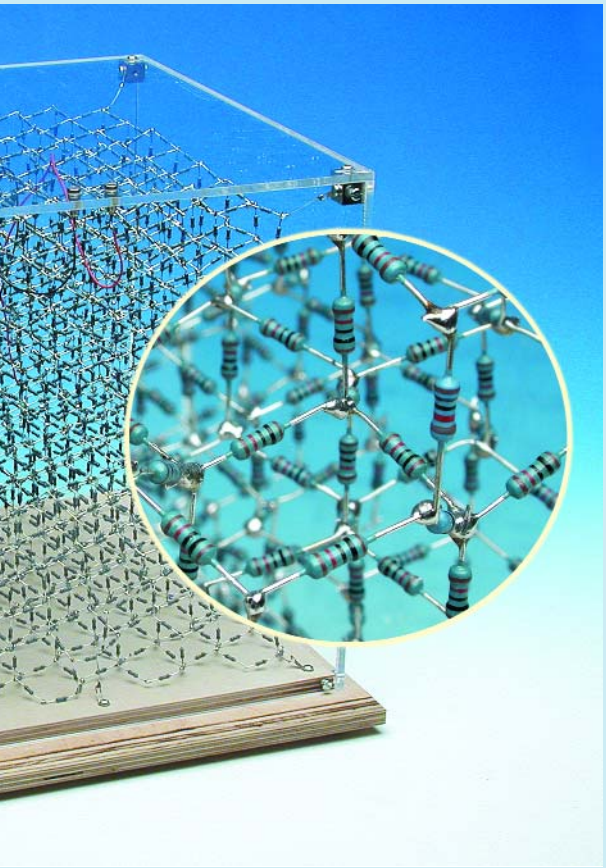


Figure 1. Infinite resistor array.

Figure 2. Ohsmann's Cube: a three-dimensional resistor array.



Quizz'away and win!

Send in the best answer to this month's Quizz'away question and win this **Voltcraft USB Oscilloscope** from **Conrad Electronics**, worth £135.

All answers are processed by Martin Ohsmann in co-operation with Elektor editorial staff. Results are not open to discussion or correspondence and a lucky winner is drawn in case of several correct answers.

Five participants with the correct (or best) answer are rewarded with a 128-MB Elektor Electronics USB Memory Stick.

Quizz'away conditions

Please send your answer to this month's Quizz'away problem, and/or the supplementary question by email, fax or letter to:
Quizz'away, Elektor Electronics, PO Box 190, Tunbridge Wells TN5 7WY, England. Fax (+44) (0)1580 200616. Email editor@elektor-electronics.co.uk, subject: 'quizzaway 09-04'.

***The closing date is 24 September 2004**. The outcome of the quiz is final. The quiz is not open to employees of Segment b.v., its business partners and/or associated publishing houses.*