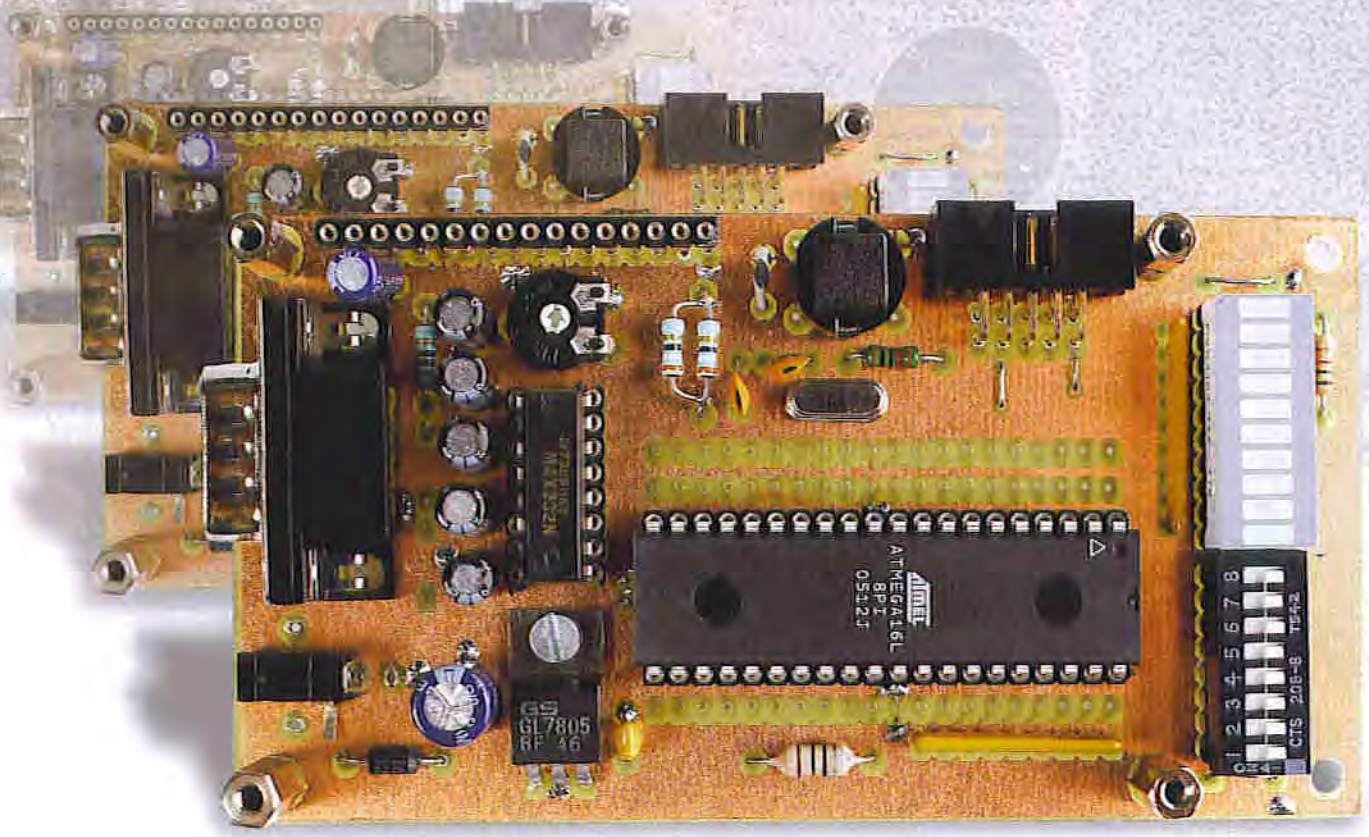
Mini ATMega Board



Florian Schäffer

It's about time, you might say! After all, even if the ATmega microcontroller isn't necessarily mega, it's still quite popular and we feel it hasn't really received the attention it deserves in *Elektor Electronics*. The Mini ATmega Board described here puts a change to that, and a nice application for it — our Standalone OBD2 Analyser— is featured in an accompanying article.

The Mini ATmega Board is a complete application board for the ATmega16 and ATmega32 members of the Atmel AVR family. It includes an LCD port, a serial interface, and abundant I/O lines. For programming, the board also has a mini-interface for communicating with a PC serial port.

There are numerous potential applications for small single-board computers in the domestic and hobby realm, such as central heating controllers, roller-shutter controllers and alarm systems. Another popular application is controlling autonomous robotic systems. In any case, developing your own application programs requires a certain amount of programming expertise, and this board with its programming interface is naturally quite suitable for acquiring that expertise. It's easy to convert user inputs into actions and responses, even with only a small program.

Simple but versatile

In its standard configuration, the board is intended to be fitted with an ATmega16. This relatively inexpensive

For ATmega16 and ATmega32

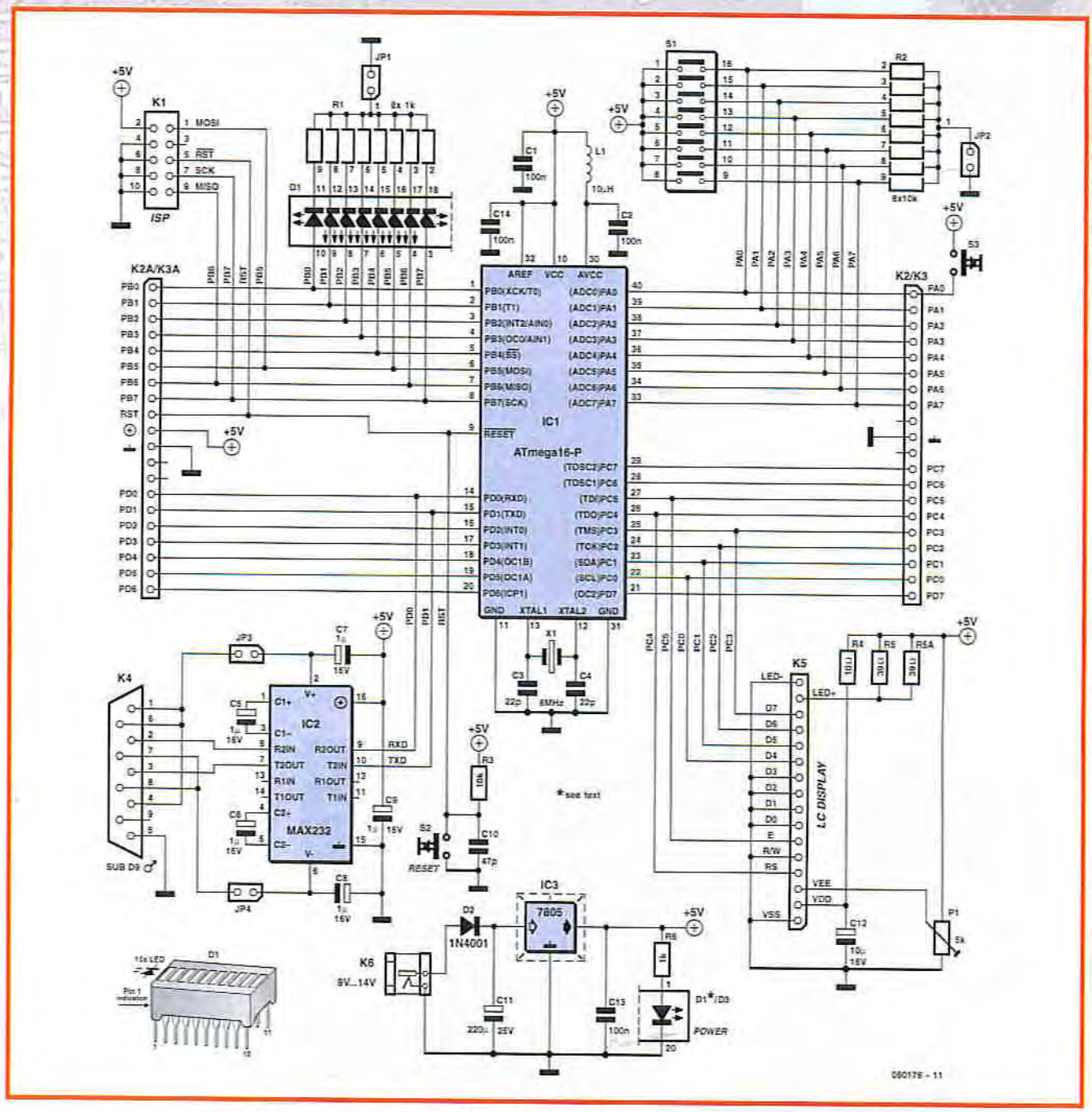
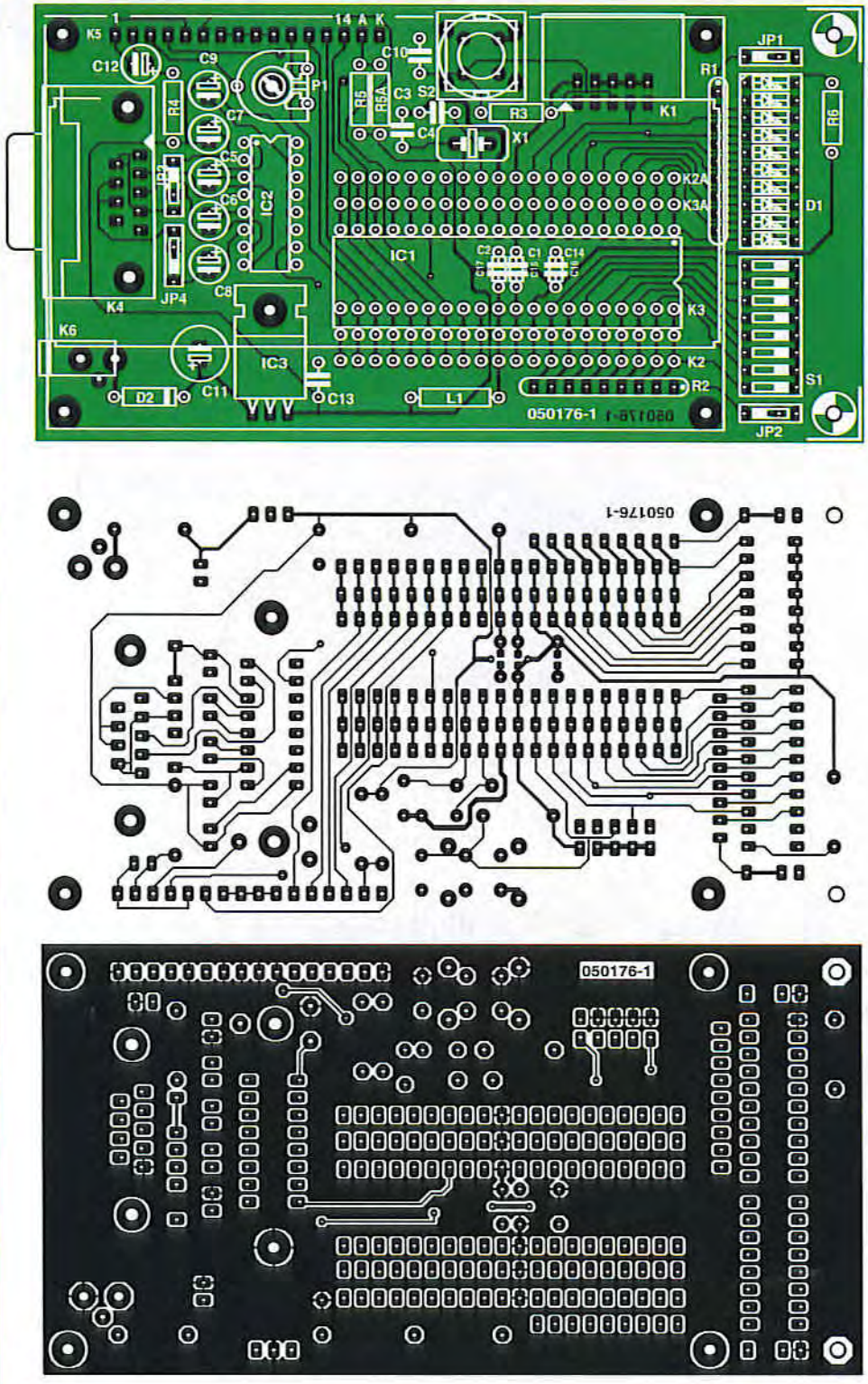


Figure 1. Schematic diagram of the Mini ATmega Board, which forms a good basis for compact applications.

microcontroller from the 8-bit AVR RISC family is quite suitable for beginners due to its DIL package, which is easier to handle than the SMD packages of the smaller AVR microcontrollers. The ATmega8, which comes in a skinny 28-pin package but still has a large number of I/O pins, is also very popular. However, its 8 KB of internal flash program memory is a bit on the skimpy side, so we prefer the larger ATmega16 with twice as much memory. If you need even more memory, the pin-compatible ATmega32 can also be fitted to the board. As the latter device is also largely software-compatible, it's generally only necessary to recompile the software in question without having to make any other modifications.

Another reason for the popularity of the ATmega processors is the fact that many major elements are already integrated. In addition, the 32 I/O lines of the two larger versions provide adequate connectivity for supplementary hardware, including eight pins that can be routed to an internal analogue-to-digital converter. The processor can run at a



COMPONENTS LIST

Resistors
R1 = SIL array 8 x 1kΩ
R2 = SIL array 8 x 10kΩ
R3 = 10kΩ
R4 = 10Ω
R5, R5A = 39Ω
R6 = 1kΩ
P1 = 10kΩ preset

Capacitors
C1, C2, C13, C14 = 100nF ceramic
C3, C4 = 22pF
C5-C9 = 1µF 16V radial
C10 = 47pF
C11 = 220µF 25V radial
C12 = 10µF 16V radial

Semiconductors
D1 = LED-Array with 10 LEDs
D2 = 1N4001
D3 = contained in D1
IC1 = ATmega 16-PC (DIP40 case)
IC2 = MAX232 (DIP16 case)
IC3 = 7805 (TO220 case)

Miscellaneous
JP1-JP4 = wire link, microswitch or 2-way pinheader with jumper)
K1 = 10-way boxheader
K2, K2A = 20-way SIL pinheader
K3, K3A = optionally fitted
K4 = 9-way sub-D plug (male), PCB mount
K5 = 16-way SIL socket, (e.g. cut off from a 40-way wirewrap socket) and mating SIL plug
K6 = mains adaptor DC socket, PCB mount
IC socket, 1 x 40 way, 1 x 20 way, 1 x 16 way
S1 = 8-way DIP switch
S2 = pushbutton, 1 make contact (e.g. Conrad Electronics # 700665)
L1 = 10 µH (fixed inductor)
X1 = 8MHz quartz crystal
Enclosure, Hammond type 1591-D blue *
16-way flatcable (a few centimetres)
PCB, order code **050176-1**, supplied including programming adapter board **050176-2**.
* not required in combination with OBD2

Figure 2. Track and component layouts of the circuit board.

maximum clock rate of 16 MHz, which yields respectable computing power. Processing complex, time-critical tasks is usually not a problem, because most of the machine instructions require only two system clocks for execution. Additional useful features include internal software UARTs for serial communication and the ability to respond to interrupts and handle system crashes using a watchdog timer.

Everything on board

As you can see from the schematic diagram (**Figure 1**), only a few components are fitted to the board. Of course, operating power and a clock source are

essential. The first is provided by a 5-V voltage regulator (IC3), and an 8-MHz crystal sets the rate of the microcontroller's internal oscillator.
Nearly all of the other components are used to support the various interfaces. K1 is the connector for the programming adapter. The circuit arrangement is compatible with conventional 10-lead adapters, such as those used with the Atmel STK boards.
The RS232 port (K4) can be programmed via the internal software USART, and it can achieve high data transmission rates thanks to interrupt control and buffering. The two jumpers JP3 and JP4 allow +10 V and –10 V to be connected to the D-Sub connector in order to power 'vampire circuits'. A null modem cable must be used to connect the port to a PC, and jumpers JP3 and JP4 should be left open in that case.

A few switches and LEDs are naturally quite welcome for simple experiments using the board. Consequently, a set of LEDs (LED array D1) and DIP switches (S1) are connected to the I/O lines. The ATmega can supply up to 40 mA per pin to drive LEDs or other components directly without additional circuitry. If you want to use the port lines for some other purpose, the pull-down resistors (array R2) and LEDs can be disabled by jumper JP2 or JP1, respectively. If port A is used as an output, the DIP switches must be switched off (open) to avoid damage to the IC.
Components L1, C2 and C14 are provided to improve the accuracy of the A/D converter. If an accurate A/D converter is not needed, these components can be omitted and L1 can be replaced by a wire bridge.
A LCD module with four rows of 20 characters (4 × 20) is the preferred choice for the display and a perfectly suitable type is offered through the Elektor SHOP. The board is laid out to exactly fit such a display module. The display can be connected using a pair of flat cables. However, a more elegant solution is solder half a wire-wrap IC socket to the circuit board. If a pin header is fitted to the LCD board, it can be plugged directly into the socket. As there are many similar versions of this standard type of display, you should check carefully to make sure the pinout matches the board layout. The modern blue versions are of course easier on the eye than the usual green versions, but as so often happens, it costs a bit more to have a special taste.

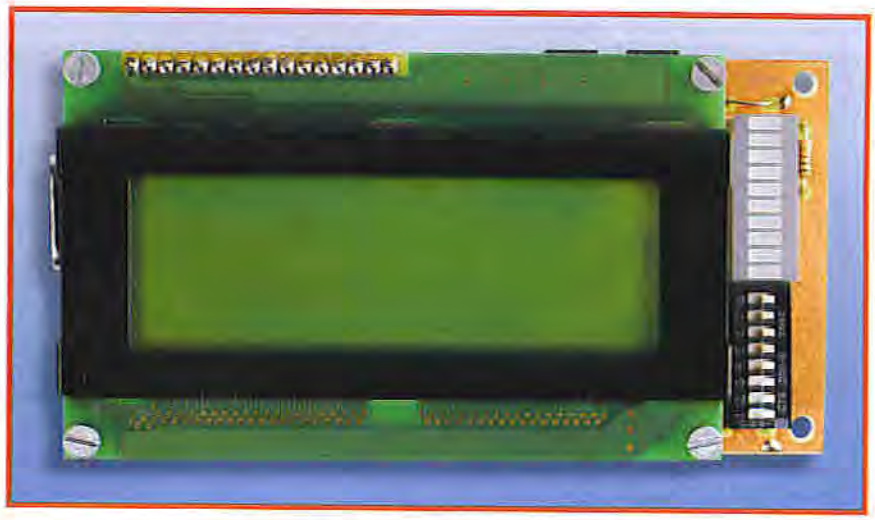
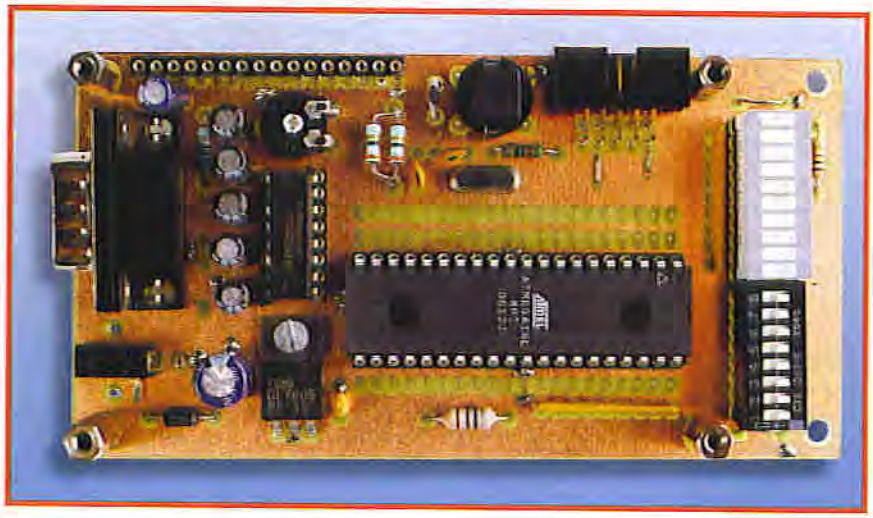


Figure 3. The fully assembled prototype board.



Don't forget to adjust the contrast setting with P1 when you first power up the board – otherwise you might think the display is defective, because you won't see anything if the setting is wrong.

A LED array with ten LEDs is recommended for D1. In that case, D3 is automatically the tenth (uppermost) LED in the array, and the ninth LED is not used. If you can only obtain an array with eight LEDs, fit it in the lower eight positions. The ninth position will then remain open, and a normal LED can be fitted in the tenth position for D3.
K3 and K3A can be used to fit pins on the bottom of the board. That will allow the entire board to be plugged into another board as a replacement for an ATmega for experimental purposes. K2 and K2A are left open for connecting I/O lines.
The only thing left to mention is the decoupling capacitors under the microcontroller. If you use a socket for the microcontroller, there will probably be enough room on top of the board to fit small capacitors. If that isn't possible, solder three capacitors in SMD-0805 packages to the pads provided for that purpose on the bottom of the board.

Programming

The microcontroller can be programmed directly from a PC via a simple programming adapter (see inset). A variety of small programs are available for this purpose. Although some of them are not especially user-friendly, they all do the job. For instance, you can use PonyProg to download fully compiled programs to the ATmega and set the most important (and crucial) fuse bits.

The fuse bits are used to configure the microcontroller, and this can lead to unfortunate situations in which you lock yourself out, for instance if you

Figure 4. The correct settings must be configured very carefully in PonyProg before the microcontroller is programmed.

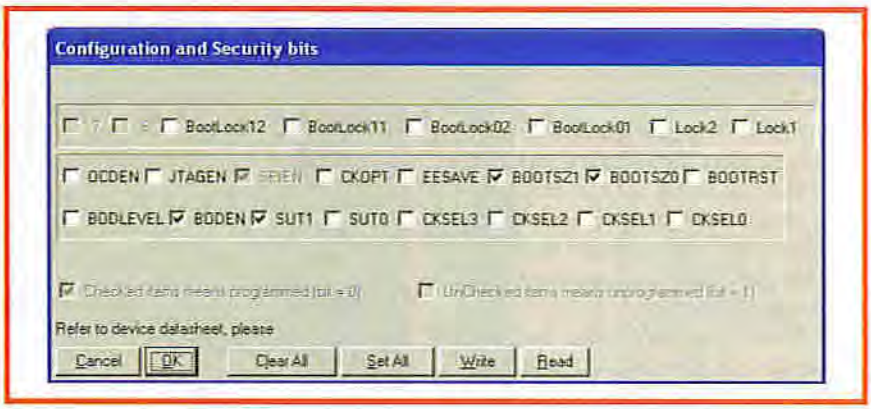


Figure 5. These fuse bits must be set to use an external 8-MHz crystal and enable brownout protection.

configure the clock source incorrectly or reconfigure the Reset pin as an I/O pin. On top of that, the AVR data sheets use a somewhat puzzling logic that is then partially inverted by the program. The first time you use the program, you must use Setup / Interface Setup to configure the right settings and then select the proper microcontroller type in the Devices menu.
It's certainly a good idea to use the programmer to read the actual settings of the fuse bits (via Security and Configuration Bits in the Command menu if you're using PonyProg) before you start programming. Not every program does this automatically. You can then compare the bit settings to the descriptions provided in the data sheet. **Figure 4** shows the required settings for using an external 8-MHz crystal and enabling brownout detection, which resets the processor when the supply voltage is applied or removed. Data loss could occur if brownout detection is not used, because no external circuitry is provided for that purpose.

Free development environment

Numerous development environments are available for the AVR family, and it's easy to develop programs in Basic using BASCOM-AVR (free for code sizes up to 4 KB), in assembly language using the free AVR Studio environment, or in GCC using the free WinAVR environment.
The example applications for this board were generated using WinAVR, compiled for the ATmega16, and converted to machine language, so all you have to do is use the programmer to download the resulting hex files to the flash memory of the microcontroller. If you want to recompile the code to use a different crystal or a different type of ATmega microcontroller, you only have to modify the 'make' file accordingly.
The AVR GCC Tutorial referenced the link list provides a very extensive introduction to C programming, although you must always bear in mind that many of the examples were developed for somewhat older types of microcontrollers (now discontinued) and require some changes. The routines for the LCD may also need slight modifications, because even though very good general-purpose templates are already available, they have not been included in the examples in order to keep the code as simple as possible.

All the examples can be downloaded from the *Elektor Electronics* website (www.elektor-electronics.co.uk) in a Zip file. You can use these programs to become familiar with all the essential components and test your board. To get started, you could use the simple 'blinking LEDs' program. The next step would be to display a message on the LED or try out data communications via the serial interface.
As already mentioned in the introduction, an interesting application for this board is described in a companion article in this issue. It is a stand-alone OBD display unit for car diagnostics, which you can also use to display vehicle-specific data 'on board' via the LCD module while you are driving.

Extensions

If you are interested in extending the board after successfully completing your initial programming and application projects, there are many possibilities – far too many to mention them all in this article. For instance, you could implement an I^2C bus with only a few additional components, add a USB port, or even connect an SC memory card to store measurement data for subsequent analysis using a PC. Naturally, you can also make contact with other ATmega fans in the various microcontroller forums, including (of course) the one at www.elektor-electronics.co.uk, and various newsgroups.

(050176-1)

Items available for this project

050176-1 PCB, supplied including programming adapter board (050176-2).

050176-42 Programmed ATMEGA16

050176-72 Kit of parts, includes ATMega board 050176-1, programming adapter board 050176-2, preprogrammed ATmega microcontroller (with OBD2 program), all components for both boards but excluding LC display.

050176-73 LCD, module, 4 x 20 characters, 60 x 98 mm, with background lighting.

050176-74 Case, Bopla Unimas 160 with perspex cover and mounting plate

Available from www.elektor-electronics.co.uk
Hyperlinks for the project.

050176-11.zip Development software including program examples (free download).

In-system programming

The minimum circuit configuration for an ATmega actually amounts to nothing more than a supply voltage. The external crystal can be omitted if you use the internal oscillator. However, the processor clock rate is not defined especially accurately in that case, so operation using the internal oscillator is only suitable for applications that are not time-critical.

Aside from the supply voltage and the crystal (if used), you also need a programming interface port to load a program into the ATmega. As the ATmega has an in-system programming (ISP) interface, it can be programmed in the application circuit – or reprogrammed as often as desired. The flash memory can handle up to 10,000 write operations, and the EEPROM can handle up to 100,000.

The programming adapter described here can be connected directly to a PC using a one-to-one cable. Connect the other end of the adapter to the ISP connector (K1) of the Mini Mega Board, and you're all set to program the microcontroller directly from your PC.

The circuitry of the programming adapter (**Figure 1**) can be built into the housing of a 9-way D-Sub cable connector to produce a handy programming adapter cable.

Assembling the circuit on the small circuit board (**Figure 2**) is not particularly difficult, but it does require a steady hand and a bit of care. Before fitting the components, check to make sure the board fits in the housing of the D-Sub connector. Then fit the SMD components first, followed by K2 on the other side of the board. Before soldering the D-Sub connector (K1), check again to make sure the board fits. In our prototype, we removed the rear wall of the connector housing. That allowed K2 to just fit, and the flat cable could be fed out without any problem. Do not solder K1 in place until you are sure that the board fits in the connector housing. It will probably be necessary to shorten the pins of K1 slightly and fit the board at a bit of an angle. As the connector housing is made from metal or conductive plastic, you should cover the inside with insulating tape to prevent unwanted contact with the circuit board.

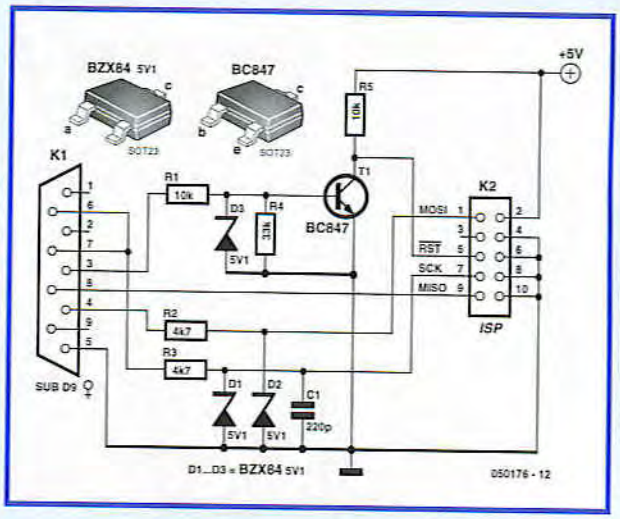


Figure 1. Circuit diagram of the programming adapter.

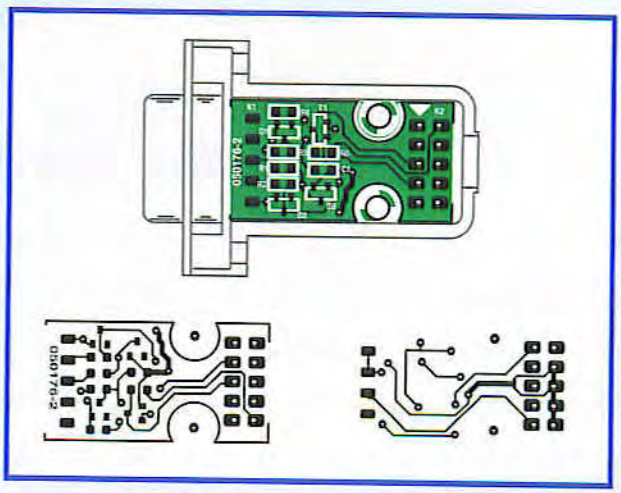


Figure 2. Track side and component side layouts of the programming adapter circuit board.

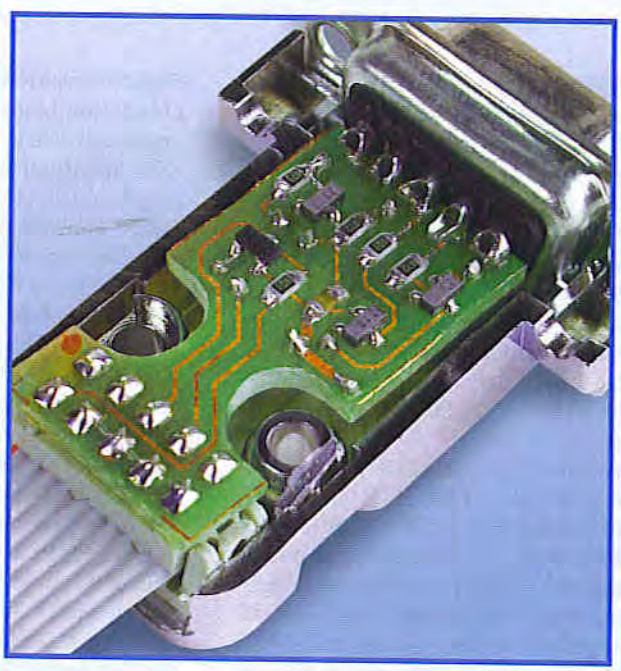


Figure 3. The circuit board with the SMD components fitted.

COMPONENTS LIST

(programming adaptor)

Resistors (SMD 0805)
R1,R5 = 10kΩ
R2,R3 = 4kΩ7
R4 = 33kΩ

Capacitors (SMD 0805)
C1 = 220pF

Semiconductors
D1,D2,D3 = zener diode 5V1, 250 mW, SOT23 case (e.g. BZX84 5V1 SOT23)
T1 = BC847 (SOT23)

Miscellaneous
K1 = 9-way sub-D socket (female) with case
K2 = 10-way pinheader
10-way flatcable (0.5 - 1 m)
10-way UIDC connector for flatcable
PCB, order code **050176-2** (see SHOP pages or website)