ELEKTOR ELECTRONICS

JUNE 2003
£3.45

THE ELECTRONICS & COMPUTER MAGAZINE

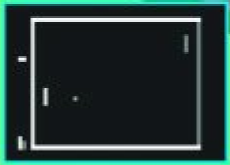
www.elektor-electronics.co.uk

Night Light Control

Cooling Electronic Components

AVR TV Tennis

back to the seventies



Electronic Knotted Handkerchief

Modifying PC AT Power Supplies

Lightning Intensity Detector

C Compilers for the PIC Micro

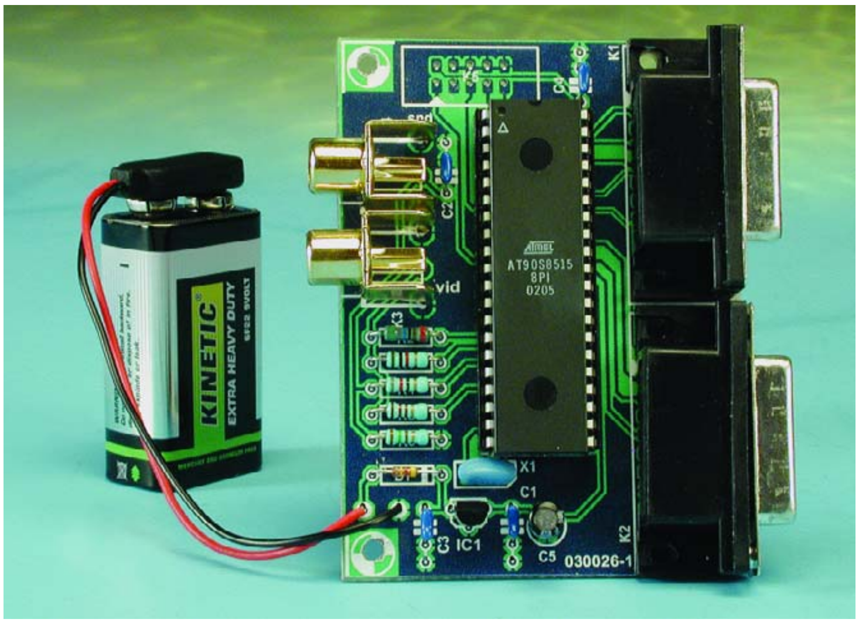
06
9 770268 451104

AVR TV Tennis

Play MahPong, the Single-Chip Edition!

Design by M. Hasenstab MAH@jkdesign.de

The circuit described here employs just a single microcontroller to put the classic MahPong video game (a.k.a. TV Tennis) on the TV screen. Guaranteed 1970's nostalgia!



In this day and age of gigahertz PCs with massive amounts of user memory, only extremely extensive games using all the latest 3-D video effects and ditto sounds appear to satisfy the younger generation. If such a game runs too slow, well that simply means your PC is sluggish (again) and needs to be replaced a.s.a.p. only to discover adverts telling you that the machine you've just unpacked from the box is… outdated! Forever gone are days of the first home computers for which highly optimised software was written that pushed computing power to levels never dreamed of when these early machines were first sold.

The challenge and excitement of the present project is in its minimalist approach to obtain, in a word, nostalgia. Today, instead of a box full of complex, purpose-designed hardware, a single 8-bit AVR microcontroller is pushed for maximum performance. The original version of the TV Tennis game, published in Elektor November 1976, contained 13 TTL ICs and 50-odd discrete parts. Follow-up articles published in 1976 added dozens more ICs that reportedly made the power supply too hot to touch, at least in the prototype used for lunchtime amusement in the Elektor lab.

Returning to 2003, the only storage capacity available on the AVR chip we'll be using is 8 kbytes of program memory and 512 bytes of RAM. The video signals are mixed together from a couple of port pins and resistors, while a Timer is used to generate the primitive sounds used in the MahPong game. During the dark (actually, blanked) periods at the top and bottom edges of the screen, the vexed microcontroller is allowed to handle the rules of the game, interpret the user input, as well as do various chores. Nothing even vaguely resembling an operating system or similar imposes itself between the object code burned into the micro and the naked silicon on the chip!

The rules of the MahPong game probably only require clarification to the younger among Elektor readers. Each player employs a bat to bounce a ball back to the opponent. If the ball gets past your bat and bounces against the wall behind it, your opponent gains one point. The score is prominently displayed in the top left hand corner of the screen. The first player to reach 21 points is the winner and is awarded a little ceremony consisting of a victory tune!

Hardware? What hardware?

The extremely simple circuit diagram shown in **Figure 1** is typical for a turnkey microcontroller application like the MahPong game. As always in these cases, daunting functions become possible thanks to ingenious software and the functional blocks available inside the microcontroller. Consequently, hardly anything appears in the way of external components.

Resistors R1, R2 and R3 between the AVR micro and connector K3, together with the 75-Ω terminating resistance inside the TV set, form a potential divider. The 90S8515 micro is capable of supplying about 18 mA through its port pins, although it must be noted that due to its internal resistances, the output voltage is rather current-dependent. If you want to calculate accurately, you first have to establish the currents required for a number of discrete signal levels to appear across the 75-Ω input impedance represented by the CVBS (composite video input) on the monitor or TV set. Such calculations yield the following results:

R1: Sync
Black level: 0.3 V
Current from Sync Pin 14:
 0.3 V/75 Ω = 4 mA
Output voltage at Port:
 4.5 V at 4 mA
Required total resistance:
 4.5 V/4 mA = 1125 Ω
Series resistor:
 1125 Ω – 75 Ω = 1050 Ω (1 kΩ)

R2: PixelOut1
Level for bright grey: 0.8 V
Current from Pin 28:
 (0.8 V – 0.3 V) / 75 Ω = 6.7 mA
Output voltage at Port:
 4.0 V at 6.7 mA
Required total resistance:
 4 V / 6.7 mA = 600 Ω
Series resistor:
 600 Ω – 75 Ω = 525 Ω (576 Ω, 1%)

R3: PixelOut0
Level for dark grey: 0.5 V
Current from Pin 24:
 (0.5 V – 0.3 V) / 75 Ω = 2.7 mA
Output voltage at Port:
 4.6 V at 2.7 mA
Required total resistance:
 4.6 V / 2.7 mA = 1700 Ω
Series resistor:
 1700 Ω – 75 Ω = 1625 Ω (1.2 kΩ)
This value causes dark grey to appear a little brighter but still distinguishable from bright grey.

Potential divider R4/R5 on the sound output of the AVR chip reduces the voltage level from 'digital' (5 V) to about 1 volt. This allows a sufficient volume range on the TV set or monitor. If you want a slightly less aggressive sound, then capacitor C2 may be connected in parallel with R5.

Digital joysticks are connected to sockets K1 and K2. A digital joystick has a switch for each direction. When actuated, each switch connects its own control line to ground. Together with resistors integrated in the AVR micro, these lines allow joystick movements to be recognised reliably.

Interface connector K5 is a 10-way box-header of the same type used on AVR evaluation boards. It allows the microcontroller to be programmed in-circuit using one of the well-known AVR programmers.

Power-wise the microcontrollers from the Atmel AVR series are marked by a rather Spartan lifestyle. The 78L05 voltage regulator in the circuit reduces the battery input voltage to +5 V. The circuit draws less than 30 mA which is easily furnished by a 9-volt battery. An 8-MHz ceramic resonator is used to help generate the microcontroller clock signal. If you happen to have a low-power stabilised 5-V supply available, then the 78L05 may be omitted from the circuit.

Powerful software

The software developed for the project may be thought of as consisting of the following modules:

Reset and initialisation routine
This module initialises the directions and levels of the I/O ports, plus organises the controller's inner life.

Main program
This bit of software not only awaits user action for the game selection, but also calls the relevant game routines while the game is being played.

Play routine PLAYPONG
This module is synchronised to the picture output and on being called first waits for the VSYNC signal that marks the start of a new picture. Once VSYNC is present, the players' input is examined to enable the position of the bats to be computed. Next, the routine searches for obstacles in the trajectory of the ball. If none is found, the ball keeps moving along a straight line. If the ball encounters an object, it bounces back into the playing field

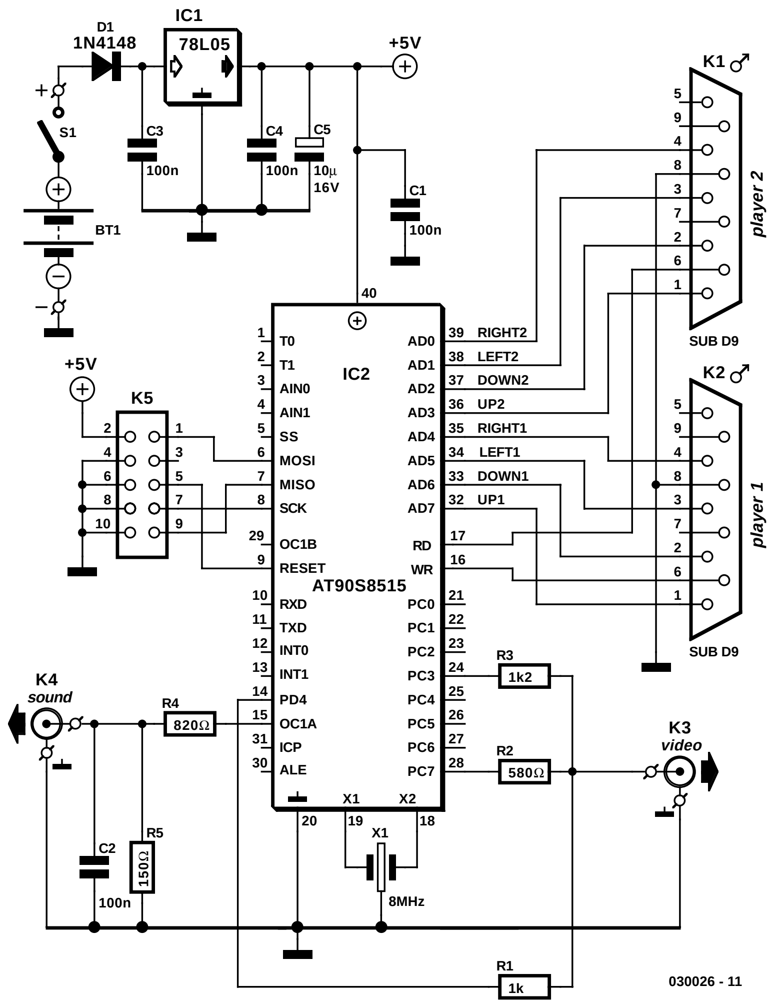


Figure 1. The minimalist approach: circuit diagram of the TV Tennis game, 2003-style.

according to the simple rule *angle of incidence = angle of reflection* and moves on in the new direction. Also, the software checks if the ball touched the left-hand or right-hand wall. If so, the score is updated. Each time the ball is in contact with a bat or a wall, an object-specific sound is generated. Once a player reaches the number of points required to win the game, a "Winner" screen is displayed and the game is ended.

TMR0 Interrupt routine
This piece of software for the video signal generating routines runs quasi in parallel with the main program. The timing of the system is crucial to the way the video generation works. For example, the video routine may never be interrupted or called with a delay. Consequently, only one interrupt routine is allowed (and that's the video generator proper), and the rest of the user program actually has its available computing time allotted by the video routine.

In addition to these larger modules we have:

Read/Write utilities for operations in the picture memory.

Sound utilities for starting sounds, playing and stopping them.

Composite video

The CVBS signal (composite video, blanking, sync) is intended to generate a monochrome image and forms the basis of extensions and standards defining colour TV, like PAL and NTSC. A CVBS picture may have 625 lines with a duration of 64 μs each. A complete picture takes 40 ms to build, which implies a frame frequency of 25 Hz. Interlacing is used to reduce picture flicker — in this system the lines of successive rasters are not superimposed on one another but are interlaced, two rasters constituting one picture or 'frame' (**Figure 2**).

The CVBS signal employs just one wire to carry synchronisation as well as brightness information. The instantaneous level of a CVBS signal is between 0 V and 1 V. The required terminating resistance of 75 Ω is contained in the monitor or TV set. The rear porch (black reference level) is defined as 0.3 V, so all levels of grey, right up to white, are comprised between 0.3 V and 1 V. The monitor interprets an instantaneous level of 0 V as the sync pulse. The horizontal sync pulse (HSYNC) lasts 4.7 μs and marks the start of a picture line. Likewise, a 160-μs long signal of 0 V is taken to mean the vertical sync pulse. Each half image consists of 312.5 lines. The timing diagram in **Figure 3** shows that VSYNC and HSYNC coincide for the first half image, but are offset by 32 μs for the second half. This allows the monitor to discriminate between the first and second half image and display them correctly.

An example of the contents of a CVBS video line is shown in **Figure 4**. The line starts with the rear porch (i.e., the invisible area to the right of the image). The appearance of the HSYNC pulse tells the monitor that a new picture line is about to start. Next comes the front porch (left-hand invisible area). This signal level is the reference for the darkest (blackest) content of the upcoming video signal. The active range contains the visible contents of the line (here, a staircase signal).

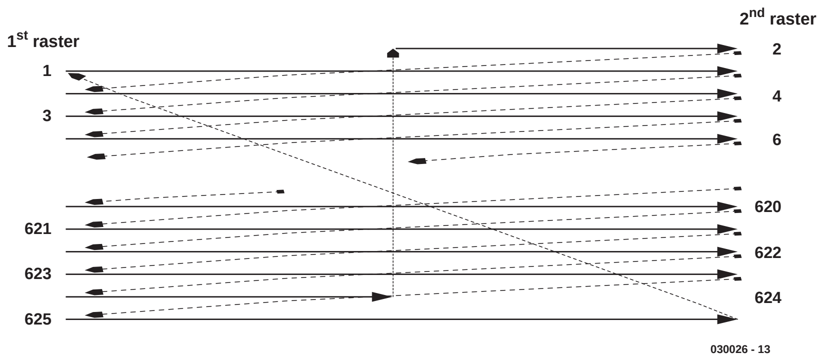


Figure 2. The CVBS video signal.

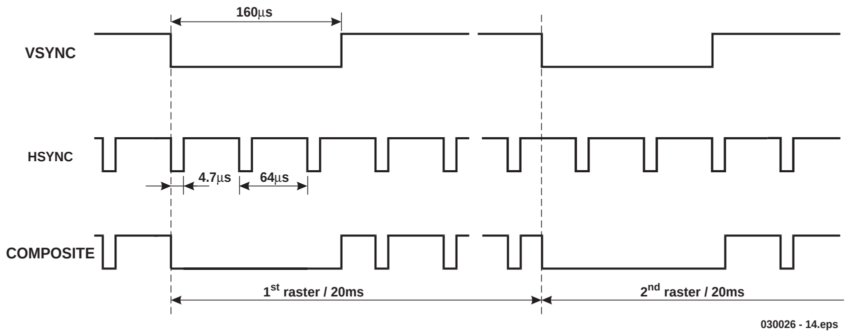


Figure 3. Composite sync signal.

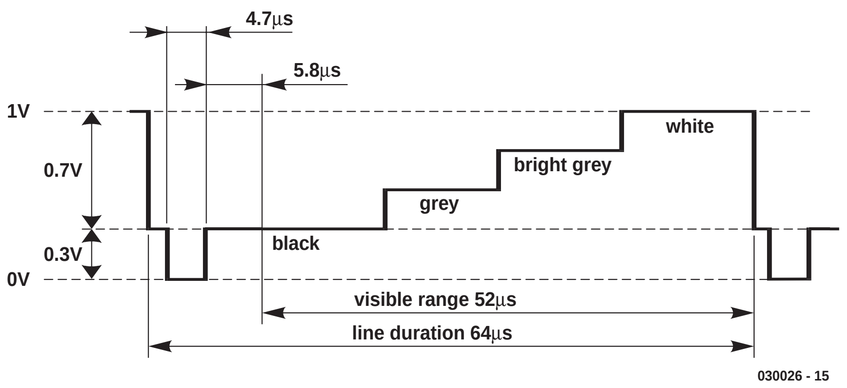


Figure 4. CVBS picture line signal.

Video signal generation inside the AVR

The graphics system, which does most of its work largely unnoticed by the user program, requires a significant portion of the available com-

```
      mov   HCOUNT, XL             ; shift Low Byte of picture memory pointer to
      subi  HCOUNT, -13            ;  HCOUNT and add 13 to value

                                   ; cycles; comment
      ld    REGA, X+               ; 2 ; Load picture data from SRAM into REGA
                                   ;   ; The following code takes 16*12 cycles: 192 cycles = 24 us
STREAMLINE_SHIFTER:                ;
      out   PIXELPORT, TMPL_I      ; 1 ; Write pixel 0 onto pixel port (PORTC)
                                       ;  note that only PC7 (bright grey) and
                                       ;  PC3 (dark grey) are used, hence
                                       ;  any values may be present on other pins.
      mov   REGB, REGA             ; 1 ; Copy REGA into register REGB
      lsl   REGB                   ; 1 ; Shift REGB 1 bit left
      nop                          ; 1 ; Idle

      out   PIXELPORT, REGB        ; 1 ; Write pixel onto pixel port
      lsl   REGB                   ; 1 ; Shift REGB left
      nop                          ; 1
      nop                          ; 1

      out   PIXELPORT, REGB        ; 1 ; Write pixel onto pixel port
      ld    REGA, X+               ; 2 ; Load new pixel data into REGA
      lsl   REGB                   ; 1 ; Shift REGB left

      out   PIXELPORT, REGB        ; 1 ; Write pixel onto pixel port
      cp    VRPOINTL, HCOUNT       ; 1 ; Compare picture memory pointer with
                                       ;  byte counter
      brne  STREAMLINE_SHIFTER     ; 2 ; If unequal, go to STREAMLINE_SHIFTER'
; end shifter;
```

Figure 5. Source code snippet.

puting power. If, however, the picture 'assembly' is assigned to an interrupt routine, the user program is simply 'pushed aside', without noticing the serious number crunching that's done in the background.

We will first have a look at the picture memory. The AVR has available just 512 bytes of RAM which, we're glad to note, is still sufficient to store 48×30 pixels with up to four grey levels (2-bit coding: 00 = black; 01 = dark grey; 10 = bright grey; 11 = white). In fact, the RAM has a spare capacity of 152 bytes for use as a stack area and for variables storage. The image memory is organised as illustrated in **Table 1**.

Counting starts in the left-hand top corner at coordinate 0/0. One byte always contains four pixels, with the High/Low values representing the brightness distributed across the two nibbles. The picture memory is cyclic (continuous).

The active (visible) part of a video line lasts 52 μs. Consequently, at a horizontal resolution of 48 pixels, only 1 μs is available per pixel. Because the AVR micro has a cycle time of 125 ns, the following tasks should be completed within eight cycles: (1) establish picture memory address; (2) load pixel brightness information from picture memory; (3) set appropriate port pins. The piece of assembly code shown in **Figure 5** arguably presents the fastest way of effectively transferring pixel data to a port pin. The code shows how different execution times of individual instructions are combined to write exactly one pixel in each 4-cycle block. This code snippet proves that the AVR micro is capable of continuously displaying a pixel every four machine cycles (500 ns). Theoretically this 'throughput' would allow a horizontal resolution of 104 pixels. To obtain the desired (lower) resolution, four NOPs (no-operation instructions) have been added to each read/write block to slow down the shift logic to 1 μs per pixel.

Of course, the vertical resolution could be the number of visible lines that appear on the display. However the available memory being limited, we need to leave this at 48×30 pixels and perhaps aim for higher resolution in a future project. Here, the pixels are squares — eight cycles (1 μs) wide and 16 lines high. The upshot is that the visible image has a width of 48 μs (of 52 μs) and a height of 480 lines (of 625). The resulting blank areas around the picture allow the game to appear reliably on most, if not all, (TV) screens and monitors.

Having solved the time-critical aspect of transferring the video signal, we still need to add the synchronisation signals to the pixel data. A look at **Figure 6** shows that during the video line the processor is fully occupied with the pixel data and has no time for other tasks. Conse-

Table I. Picture memory contents

```
BYTE0:       H00,00 H01,00 H02,00 H03,00 L00,00 L01,00 L02,00 L03,00
BYTE1:       H04,00 H05,00 H06,00 H07,00 L04,00 L05,00 L06,00 L07,00
...
BYTE359:     H44,29 H45,29 H46,29 H47,29 L44,29 L45,29 L46,29 L47,29
```

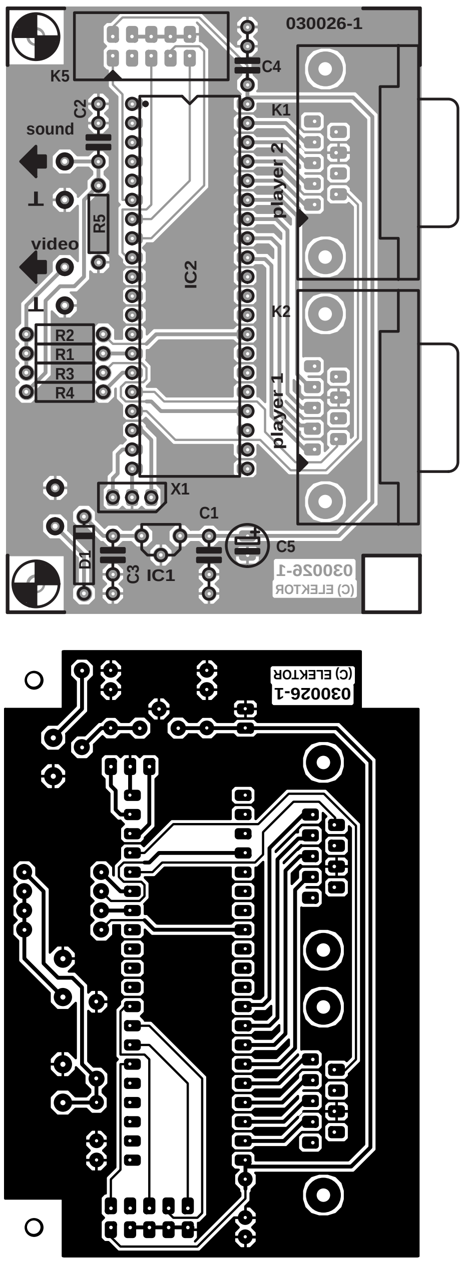


Figure 6. Single-sided board for the game.

COMPONENTS LIST

Resistors:
R1 = 1kΩ
R2 = 576Ω, 1%
R3 = 1kΩ2
R4 = 820Ω
R5 = 150Ω

Capacitors:
C1-C4 = 100nF
C5 = 10µF 16V radial

Semiconductors:
D1 = 1N4148
IC1 = 78L05
IC2 = AT90S8515, programmed, order code **030026-41**

Miscellaneous:
X1 = 8MHz ceramic resonator
Bt1 = 9V battery (6F22) with clip-on leads
K1,K2 = 9-way sub-D plug (male), PCB mount, angled pins
K3,K4 = cinch socket
K5 = 10-way boxheader
S1 = on/off switch, 1 contact
Enclosure, e.g., Hammond 1591BT
PCB, order code **030026-1** (main board)
PCB, order code **030026-2** (pushbutton board)
Disk, AVR source code, order code **030026-11** or Free Download.

quently the program control during the image assembly is landed with the TMR0 interrupt service routine (T0-ISR). However, when generating the black lines at the top and bottom of the picture, T0-ISR actually generates just the HSYNC signals. On completion, T0-ISR is left and the program control is given to the main program for about 59 µs. Because the timer is programmed for 64 µs (i.e., a full line), T0-ISR is called in time for the next HSYNC.

Unfortunately, when calling T0-ISR we have to take into account the so-called interrupt latency. In practice, this means that an interrupt can not cause the CPU to actually stop the execution of machine code at any instant. In fact, the current machine code will be finished to completion, the address of the next instruction is saved on the stack (to act as a return address) and only then will the CPU jump to the start of the ISR. That the AVR micro has instructions with different execution times is a complicating factor. The calling of the ISR can be delayed by an amount of four to seven machine cycles. Because our pixels have a width of eight cycles, the upper picture line would vary in length by up to half a pixel width. Such jitter is prevented by staying inside the interrupt routine from the start of the last two black lines before the visible (half) picture up to their end. All times within this range are mutually dependent using accurately adjusted instruction execution durations.

Our active video image consists of 480 lines, a complete CVBS image, of 625. Four lines are used to compensate the interrupt latencies. Of the 40 ms it takes to build a picture, about (625–480–4) × 59 µs = 8319 µs is available to the main user program.

For picture memory access the user program has three main functions at is disposal. Using PLOTSCREEN an image contained in Flash memory (title screen or empty playing area) may be copied into the graphics memory. The function SETPIXEL allows a pixel to be given a certain shade of grey in order to draw the ball or the bats on the screen. GETPIXEL, finally, enables the grey level of any pixel to be sampled. This enables, for example, collision of the ball with an obstacle to be recognised.

Sound generator

The project includes a simple sound generator that employs the sound output of the Output Compare A (OC1A) pin of 16-bit timer 1. If the counter state matches the value in compare register A, the sound output is inverted and the counter gets reset to 0. The sound system contains a expandable list of musical notes (well, frequency/length information).

Once during the assembly of a half image, that is, exactly at 20-ms intervals, the user program calls the function DOSOUND, which compares a VSYNC counter with the length of the currently played note and, if necessary, moves on to the next note. Functions STARTSOUND and STOPSOUND allow new musical 'pieces' to be started or stopped at any time.

Free Downloads

AVR microcontroller software (source code). File number: **030026-11.zip**
PCB layouts in PDF format. File number: **030026-1.zip**
www.elektor-electronics.co.uk/dl/dl.htm, select month of publication.

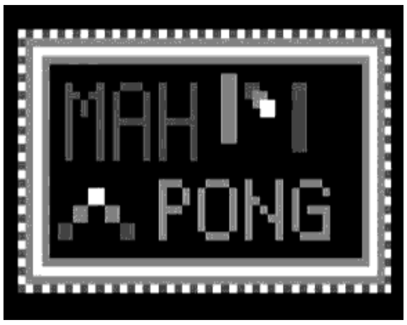


Figure 7. TV Tennis welcome screen.

Construction

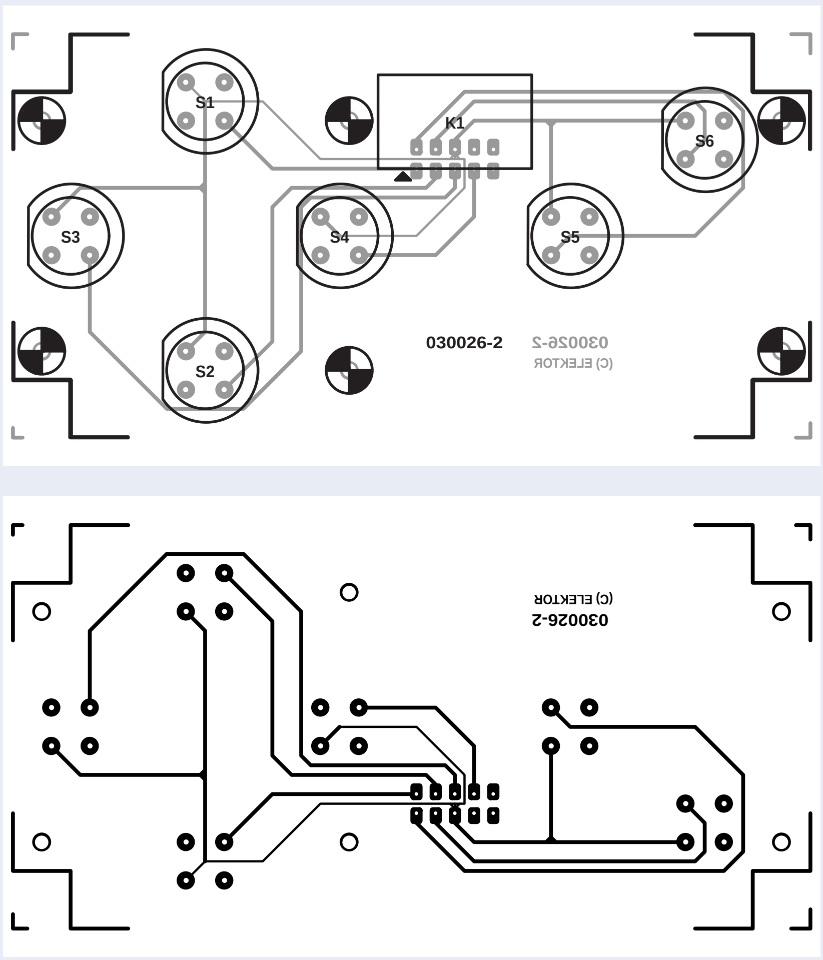
The circuit is built on a single-sided PCB of which the design is shown in **Figure 6**. With so few parts involved, the construction of the AVR TV Tennis project is unlikely to preset problems and you should be able to build the game in less than half an hour. The only two points to note are that he microcontroller should be fitted in a good quality IC socket, and that connector K5 may be omitted if you do not have the equipment to program the AVR controller 'in-circuit'.

After a thorough inspection of your solder work, the digital joysticks are connected up and the video/sound cable to the TV set (SCART or composite video input) is plugged in. The title screen (**Figure 7**) should appear immediately when the supply voltage is applied. Joystick #1 even allows a game variant to be selected — if you press the right-hand pushbutton, you play TV Tennis. The left-hand button, however, allows the bats to be moved horizontally, too, while the ball will move much slower. Although the resulting game has been called Soccer by the author, admittedly the resemblance with the real game is very remote indeed. Whichever game is played, it is started by both players moving their joysticks upwards. If the ball bounces against one of the vertical walls, one point is gained by the player at he opposite side. The first player to collect 21 points is awarded the 'Winner' screen, which may be left by both players moving their joystick downward. Have fun!

(030026-1

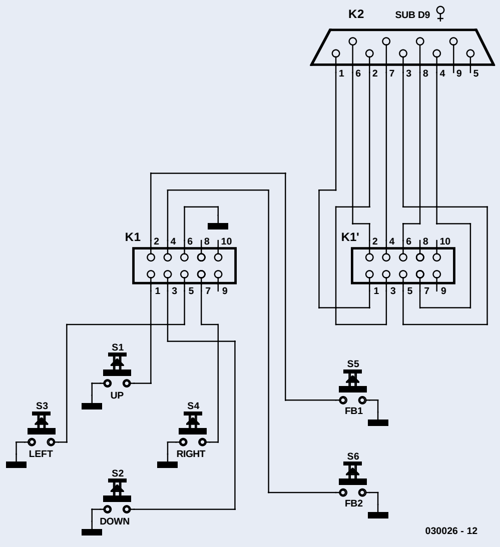
Software extensions

Only the main program and the play routines are specifically for the MahPong game. The remaining modules in the software may also be used for other games and projects. For example, the free port pins on the micro should allow serial communications to be implemented without problems. Alternatively, one more joystick could be connected up, or a digital thermometer. When an AVR controller with ADC inputs is applied, for example, the AT90S8535, it is even possible to connect analogue (PC) joysticks. The pots in these joysticks are then used as part of a R-C network, allowing the bat positions to be determined by means of the charge/discharge times. When the ATmega32 is used (AT90S8515-compatible, but 16 MHz and 2 Kbytes internal SRAM), then a significant increase in the screen resolution is within easy reach.



Digital joysticks

Digital joysticks, very common in the Commodore C64 age, are currently hard to get. If you fail to obtain a pair despite visits to second hand computer stores, flea markets, car boot sales and radio amateur rallies, you should not despair of ever being able to play an exiting game of TV Tennis. The good news is that a digital joystick is easily made at home, the circuit consisting of no more than four Direction buttons, two Fire buttons (not used by TV Tennis) and a connector. We have even designed a small printed circuit board for this mini project — designed to fit perfectly in a Hammond 1591BT case.



Cooling Electronic Components

Some really cool developments

By Reg Miles

regmiles@lycos.co.uk

One thing has characterised the electronics industry since the change from valves to solid-state devices — both components and products have got smaller. The inevitable consequence of this is that components are running much hotter, but have less space for cooling to take place. This article looks at some of the recent developments and innovations in cooling techniques.

The heatsink is no longer the complete solution that it once was, but it is still widely used. The efficiency of any design depends on the surface area of the fins, and the heat transfer coefficient (how effectively heat can actually be removed from its surface). The trouble with using air to do so is that it is a rather good insulator. But it does have obvious advantages — particularly for consumer uses.

Conventionally, heatsink fins are thin and flat. Recently, however, folded fins have been introduced made of corrugated metal sheet bonded, brazed or soldered to the heat spreader base.

This design was originally introduced for military and aerospace use, where the large surface area and light weight were advantageous, and it has spread rapidly. For omnidirectional airflows, the cross-cutting of flat fins into short, peg-like sections gives a c. 20% improvement by comparison with bidirectional types. Augmented fin techniques either add a curvature to the leading and trailing vertical edges of the divided fins (bent fin) or split and bend the fins to give the shape of a tuning fork to create a degree of turbulence; this improves heat transfer by breaking up the slow moving layer of air at the surface of the fins caused by friction between them and the air particles - a region known as the boundary layer. Pin fins (round, elliptical or hexagonal columns) achieve the same effect - and can still be used effectively with natural convection and slow moving air.

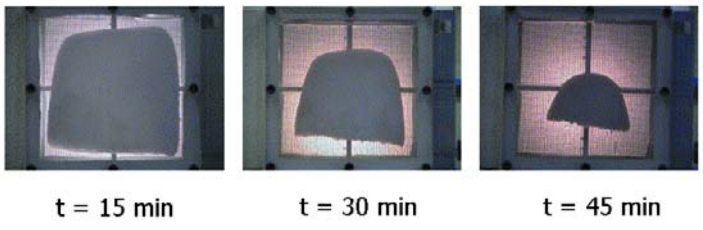


Figure 1. Visualisation of phase change energy storage (courtesy Cooling Technologies Research Consortium).

Most heatsinks are made of aluminium. Copper is also used, when the superior conduction outweighs the increased weight and cost. A recent alternative is aluminium silicon carbide (AlSiC); which is light, strong and conductive, although expensive at present.

Natural graphite is another newcomer: this has the conductivity of copper, at less than a quarter of its weight. Pyrolitic graphite and graphite fibre-based materials have still better conductive properties; but to achieve them requires temperatures of over 3000 °C, which is costly and limits them to aerospace applications at present. Graphite materials, incidentally, exhibit strong anisotropic properties, with different conductivity along axes in different directions — which limits the range of applications.

Heatsink compounds

In order to maximise the conduction from component(s) to the base of the heatsink it is usual to apply some form of thermally conducting compound that fills in any unevenness in the materials, so that there is no insulating air trapped between them. The most common is a silicone-based thermal grease; but there are a variety of other compounds available. One recent development is the use of a phase change material (PCM). This is solid up to a predetermined temperature, and then liquefies as the temperature exceeds that — spreading out and filling the interface.

PCM is also used to prevent components from exceeding a particular operating temperature, or to cool components that operate transiently. Its expansion as it changes phase and liquefies absorbs the excess heat (see **Figure 1**); and when the temperature drops the PCM will solidify again in readiness for the next temperature rise. Heatsinks filled with PCM are available, and PCM reservoirs that can be attached to PCBs and individual components.

Fans and jets

When components generate a lot of heat, then a fan is probably the answer. Or, a more recent development, the jet actuator. A device that consists of a plenum chamber and a small, electromagnetically driven diaphragm that sucks air in and blows it out again. This pulsed delivery helps to overcome the boundary layer problems described before, because the layer will thin between blasts. A small version of this, known as a Micro-Jet Array, has recently been developed which can be placed immediately below the component to blow air over it at a velocity of around 70 km/h.

Thermosiphon et al.

An even more effective method of avoiding hot spots is to conduct the heat away from the component to where it can be cooled near the outlet (or even outside the enclosure, if that is practicable); and this can be achieved by using a thermosiphon or a heat pipe — without any power being consumed. A thermosiphon is an evacuated sealed tube, typically made of copper, containing a small amount of fluid —- usually water. Being in a vacuum the fluid readily vaporises when heat is applied, enabling it to absorb a large amount of heat, the vapour rises to the other end of the pipe where the heat is conducted away and the vapour condenses on the inner surface of the tube and runs back down for the cycle to be repeated. It only requires a slightly lower temperature for the vapour to condense, so it is very efficient.

A heat pipe adds a porous wick lining on the inner surface of the tube that recirculates the liquid by capillary action to the hot end. The finer the pore structure of the wick the more the capillary action can overcome gravity: grooved and fine screen wicks will only cope with the evaporator slightly above the condenser, whereas sintered metal powder wicks can cope with any orientation.

Thermosiphons and heat pipes are becoming increasingly attractive because of their efficiency and their passive operation. They are also small: typically 3-4mm in diameter, with thinner, micro, versions now coming into use; and in lengths to suit the applications. A mobile phone, for example, would require only a short pipe; and would probably use the antenna as the condenser.

In a notebook computer it would be somewhat longer; and use either an aluminium plate under the keyboard or a heatsink as the condenser. Multiple heat pipes can also be used to carry heat from several components to the one heatsink.

A recent variation on the thermosiphon and heat pipe is to make them into a loop. The vapour travels from the evaporator to the condenser and the condensed liquid travels through a continuation of the pipe back to the evaporator. The advantages of the loop arrangement are that the vapour and liquid lines can be flexible, and can be a lot longer than a conventional pipe — more than a metre. A further variation is the pulsating heat pipe, with an internal serpentine channel in which expansion and contraction due to vaporisation and condensation set up a pulsating motion that pushes the vapour to the cold end and the liquid back to the hot end.

Temperature vs. size

The problem of cooling electronics components with increasingly high operating temperatures is being exacerbated in many cases by their smaller size. This is particularly true in applications such as telecommunications, where the heatsinks needed to cool devices like RF amplifiers are larger than what they cool. The result is a hot spot directly over the device due to ‘thermal spreading resistance’ — only that part of the heatsink is doing its job. The fan size and speed can be increased to deal with it, or aluminium can be replaced by copper or some more exotic material (chemical vapour deposited (CVD) diamond is being used for heat spreaders where the temperature is critical). But these alternatives have their disadvantages in noise, weight and cost.

An attractive solution is to embed heat pipes in the base of an aluminium heatsink. This overcomes much of the resistance, and spreads the heat fairly evenly. An even more effective solution is to use a vapour chamber, which is a vacuum vessel with a wick lining that works on exactly the same principle as the heat pipe: wherever heat is applied the fluid in the wick at that point is vaporised; and wherever the vapour comes into contact with a cooler part its latent heat is released and it condenses back into the wick. A DARPA (US Defence Advanced Research Projects Agency) project, led by Florida International University, is developing a similar heat spreader; but containing a piezo-driven micro pump to transfer the fluid to a heat exchanger, eventually to be integrated into a single module.

When there is no practical or aesthetic objections, a cold plate (or water block) makes a very effective alternative to air cooling, heat pipes and vapour chambers. This is moving into the realms of computer overclocking, where the motivation is not to prolong the reliable life of the component, the CPU, but to push it to its limits. Although, of course, it is a perfectly valid means of prolonging the reliable life of any components that do get very hot.

The cold plate may be just a container for the liquid (probably water) or have a liquid carrying pipe embedded in it. Depending on its design, electronic components may be attached to one or both sides. The system, at its simplest, would consist of cold water going into the cold plate through a tube from a supply and out through another tube. A more likely arrangement is the recirculating one, comprising a cold plate, a reservoir or expansion tank, a pump, and a heat exchanger (probably a tubed radiator — a small version of the type used in cars) with a fan to provide air cooling (a liquid to air system). Warm liquid from the cold plate goes to the reservoir, then to the pump, on to the radiator where it is cooled, and back again to the cold plate. If the cold plate has pipes embedded in it these can be fed in series or in parallel: in the former case starting at one end and snaking along the plate from side to side; in the latter case having a number of pipes going straight across from a main supply pipe on one side to a main return pipe on the other. Kits have already become available: Maplin has one (www.maplin.co.uk); and Koolance has a range of kits, and water cooled PCs (www.koolance.com). Figure 2 shows their CPU-200 Cooler installed.

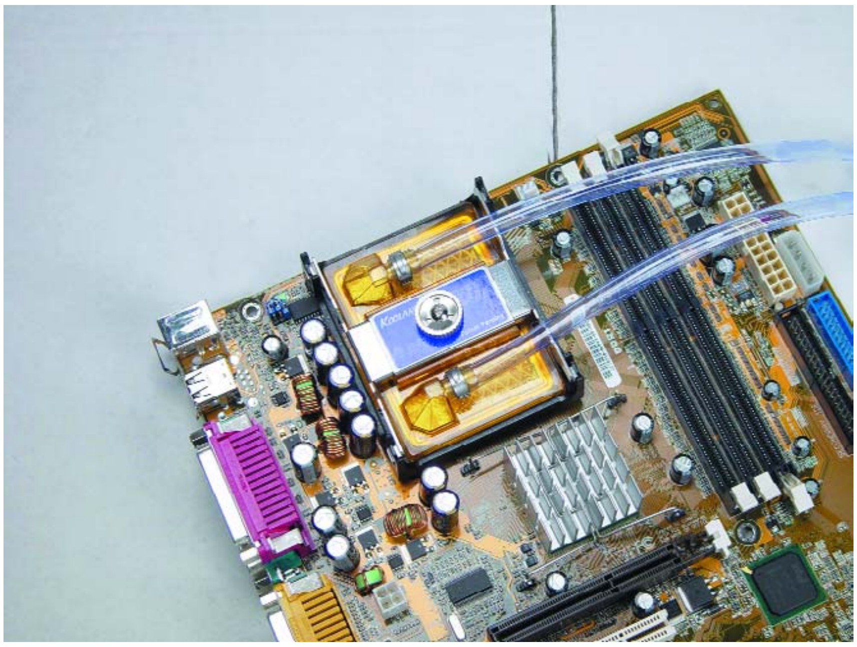


Figure 2. Koolance CPU cooler installed (courtesy Koolance).

Liquid cooling — the future?

Liquid cooling is very suitable for combining with a thermoelectric cooler (TEC) — also known as a Peltier element. This consists of two semiconductor elements, primarily bismuth telluride, heavily doped to create n-type and p-type couples (see **Figure 3**). At the cold junction, heat from the component is absorbed by electrons as an electric current raises them from a low energy level in the p-type element to a higher energy level in the n-type element; at the hot junction, electrons move from n-type back to low energy p-type and the heat is expelled. The flow of electrons is maintained by the DC current, which goes up one couple and down the other, but the carrier current, and heat, go down both. Normally, the n-type and p-type couples are combined in various numbers to produce a module, where they are connected electrically in series and thermally in parallel. A module can be small, but still produce a typical temperature difference of 70 degrees Celsius. For still greater cooling the modules can be cascaded.

Superlattice block

A new development from the Research Triangle Institute promises to give the TEC a considerable boost. It uses stacks of thin films of two alternating semiconducting materials to control the transport of phonons and electrons in the superlattices: the p-type Bi2Te3/Sb2Te3 superlattices block the former (to prevent heat being conducted back) and transmit the latter. It is claimed to be 2.4 times more efficient than a conventional TEC; and responds 23,000 times faster, because of its thinness.

Apparently, dots of the material applied to just the hot spots on an electronic component would be more effective than cooling the whole device — and save on power. Meanwhile DARPA is funding research into integrated thermo-electric-fluidic refrigeration plates, to replace the more expensive and less efficient separate TEC and coolant — a project led by CFD Research Corporation.

A variation on the TEC is the thermionic cooling device. This employs two materials separated by either a barrier layer of around one micrometre thick or a vacuum gap of a few nanometres: as the electrons absorb heat and become more energetic they tunnel from the cold side (emitter) to the hot side (collector), aided by a voltage bias — and the barrier layer or, especially, the vacuum gap prevent phonons from returning. The low voltage means that cooling can be achieved without unwanted heating. The main problem with the development of the type using a vacuum gap is that of obtaining a gap of consistent size across an area measured in square centimetres — in the laboratory, and then in production.

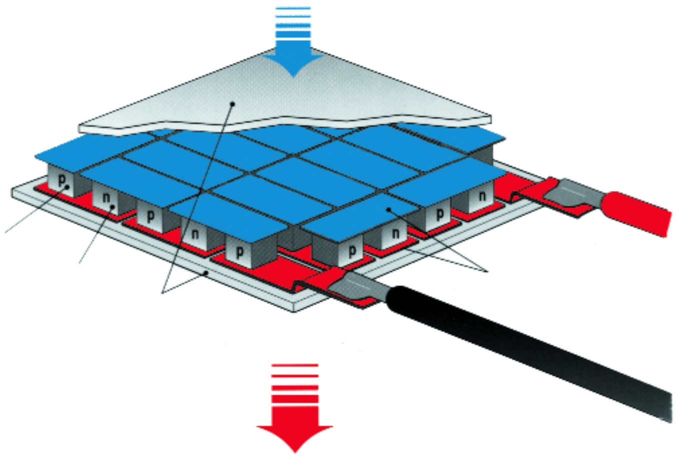


Figure 3. Basic structure of a thermoelectric cooler (TEC) (courtesy Melcor).

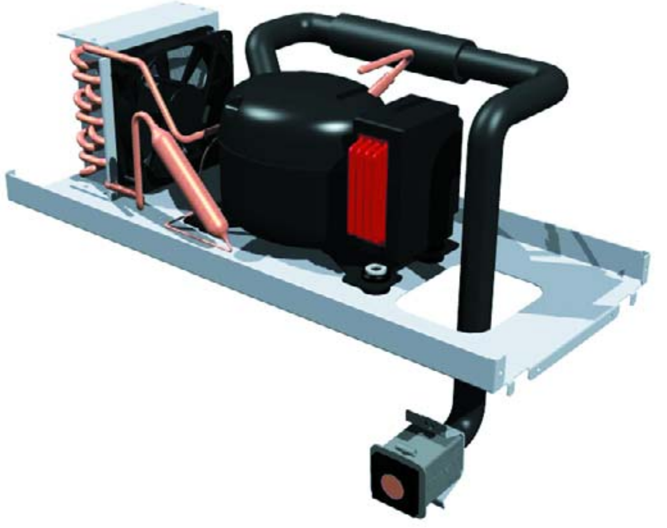


Figure 4. Vapour compression cooling system for home constructors (courtesy Vapochill/Asetek)

The fridge approach

Returning to liquid cooling, such a system can make use of conventional refrigeration to cool the liquid. In this chilled liquid (or liquid to liquid) system the refrigerator will replace the heat exchanger. Again, water is likely to be the circulating fluid. If it is required to cool at sub-zero temperatures it can be mixed with antifreeze — which does reduce the excellent heat transfer properties of water and increases the viscosity, but not significantly so. As an alternative, the system can be a refrigerator in its own right, which is normally referred to as vapour compression cooling. VapoChill (www.vapochill.com) has a range of workstations, power PCs, and a kit for home constructors (see **Figure 4**). To save space a capillary tube is used to reduce the pressure prior to the evaporator mounted on the CPU, otherwise it is the same as a normal refrigerator. Such systems are intended to be used at low temperatures, in the range from 20 °C down to –40 °C. Because it is the refrigerant itself that is being circulated, the flow rate can be less than with chilled liquid cooling. A Stanford University led project for DARPA is investigating the use of a micro cooler, using an electrokinetic pump and a micromachined evaporator incorporating a temperature sensor; with the long term goal of developing refrigeration devices based on novel compressors for the vapour phase using electrokinetic technology.

Operating CMOS devices at low temperatures gives a significant increase in performance. This is mainly because of an exponential reduction in leakage currents and an increase in transistor switching speed. The latter results from an improvement in mobility, which increases as the temperature falls because thermal vibrations in the semiconductor crystal lattice are reduced and, with it, electron-phonon scattering that slows the carrier velocity (transistor switching speed being proportional to the mean carrier velocity in the device). However, devices can fail due to mismatches in the thermal expansion of their materials, or electronic failure due to hot carrier effects — they must be suitable for use at those low temperatures.

Another barrier to using chilled cooling is the condensation that will form when warm, moist ambient air meets the cold parts of the system. All parts of the cooling system that are inside the case must therefore be insulated to prevent condensation forming.

One refrigeration technology that looks particularly promising for cooling electronics (and there are several alternatives to the familiar vapour compression system) is the thermoacoustic refrigeration device (TRD), or thermoacoustic cooler (TAC). This uses standing acoustic waves in an enclosed cavity to generate the mechanical compression and expansion of a pressurised gas (normally helium), in conjunction with a thermoacoustic core consisting of a porous stack of plates. Its operation is in the manner of a pump: expanding the gas enables it to absorb heat, while compressing it expels that heat; as the gas particles oscillate back and forth they cool down the stack by absorbing heat as they pass through it in one direction and heat it up as they travel back in the opposite direction — resulting in temperature gradient across the stack. This is exploited by having a cold and a hot heat exchanger at either end through which circulating fluid passes (water, with or without antifreeze).

Because the technique has the potential for high efficiency operation without the need for cooling liquids or mechanical moving parts (except for the acoustic driver), it is suitable for miniaturisation. DARPA has a project under way, led by Rockwell Science Centre, to develop a microelectromechanical system (MEMS) version of it; utilising piezoelectric materials and transducers for high efficiency acoustic generation. And a related project, led by the University of Utah, to integrate it with microelectronic circuits. An additional advantage of the TRD is that the degree of cooling can be controlled by adjusting the amplitude, rather than by switching it on and off like a normal refrigerator.

Watch your calories

In addition to refrigeration systems using liquid or gas to achieve cooling, research continues into systems that employ the magnetocaloric effect. In this a magnetic field is cyclically applied to a paramagnetic solid that makes it expel absorbed heat when the field is on, because all the electron spins are aligned and entropy (and its capacity to hold heat) is reduced; and lets it absorb heat again when it is off, because the electron spins are randomised again — increasing its heat absorbing capacity.

When the refrigerant (which can be water with antifreeze, or any other suitable liquid) is pumped into contact with the paramagnetic material while the magnetic field is off, its heat is absorbed, and the cold liquid goes on to cool the electronic component(s); the warmed liquid then goes back to the material (with the magnetic field on) and carries away the previously absorbed heat, and then on to a fan cooled heat exchanger. From where the cycle is repeated.

Until recently, it required superconducting magnets to work. Then researchers at the Ames Laboratory working in partnership with the Astronautics Corporation developed a refrigerator using a permanent magnet, and a rotating disc containing an alloy of gadolinium, silicon and germanium (GdSiGe) as the paramagnetic material. Nanocomposites have also been proposed to further increase the magnetocaloric effect.

A variation on magnetocaloric refrigeration is electrocaloric refrigeration, which uses electric fields to achieve the same effect. This was originally developed by researchers in Russia.

It has the advantage that electrical fields are easier and cheaper to develop than magnetic fields. Research into paraelectric materials continues, to find the optimum combination with which to achieve realistic temperature changes.

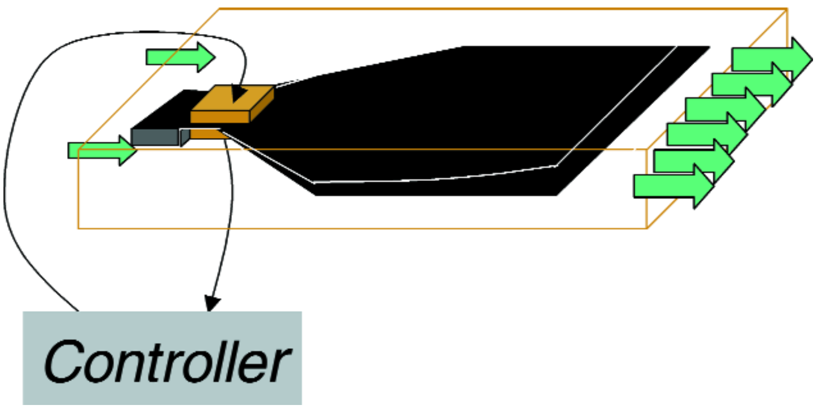


Figure 5. Principle of 'flapping' piezoelectric fans (courtesy Cooling Technologies Research Consortium).

Other developments

Just for completeness, there are a couple of other methods of liquid cooling that are used for specialised purposes. One is spray cooling, where, as the name implies, components and circuit boards are sprayed. A refinement on this has recently been announced by researchers at UCLA (University of California Los Angeles): they have developed micro sprays that cool individual components, using a matrix of nozzles 35 micrometres in diameter. These are made by reactive-ion etching; a process that gives a particularly smooth internal wall to minimise clogging by trapped contaminants. The sprayed water cools by both thermal convection and evaporation. The nozzle matrix was tailored to the distributed heat of the component, which was coated with Parylene-C, a conformal polymer with excellent dielectric properties. DARPA is funding similar research, led by Carnegie Mellon University. In this case the atomised droplets (of dielectric fluids) will be created by swirlers and vibrating piezoelectric transducers, as well as by micro nozzles. It is intended to use integrated software on the chip to control droplet sizes, spray frequencies and spray locations, based on sensing the temperature, thermal gradients and film thickness.

NASA has also been experimenting with spray cooling; and liquid immersion, with the components in a liquid bath. This will enable NASA to make increased use of commercially available components in space: by allowing them to be decoupled from the chassis, and providing radiation shielding, in addition to good thermal management.

The Cooling Technologies Research Consortium at Purdue University in Indiana was originally founded in 1999 by Suresh Garimella, a professor of mechanical engineering at Purdue, and now has ten consortium members: Aavid Thermalloy, Apple, Delphi-Delco Electronics, Eaton Corporation, General Electric, Intel, Modine Manufacturing, Nokia, Rockwell Automation and Sandia National Laboratories. In addition, there are two Consortium Supporters: Johnson Matthey and Philips Research.

"Industry comes to us with a technical problem, and we conduct research to help solve those problems.", Garimella said.

A number of projects are under way, one of which is investigating micro channel heatsinks. These have micrometre-sized channels that carry a cooling liquid; and have many times the heat transfer coefficient of conventional liquid cooled heatsinks.

The CTRC is investigating the characteristics of heat transfer and fluid mechanics in micro channels, which differ in many respects from conventional designs because of the small scale.

Another project deals with piezoelectric fans — developed by Garimella and Arvind Raman, an assistant professor of mechanical engineering at Purdue. These have piezoceramic patches bonded onto thin, low frequency flexible blades to drive the fan at its resonant frequency — alternating current causes the ceramic to expand and contract, giving a flapping motion (**Figure 5** shows the principle). The technology is attractive because the fans can be small: small enough to fit into a mobile phone or portable computer; and smaller still to fit onto a chip and cool it directly, with blades only 100 micrometres long. They also consume only 2 mW of electricity, compared to about 300 mW for a conventional fan. And, without magnets, there is no electromagnetic noise to interfere with signals. The Consortium is also investigating and improving flat heat pipes and phase change materials.

Conclusion

Cooling used to be almost an afterthought, and something of an art when it was applied. But the latest hot-running components have put an end to that casual approach. It is now essential to consider the needs at an early stage in the design process, and research is turning its application into a science.

(020411-1)

Addendum:

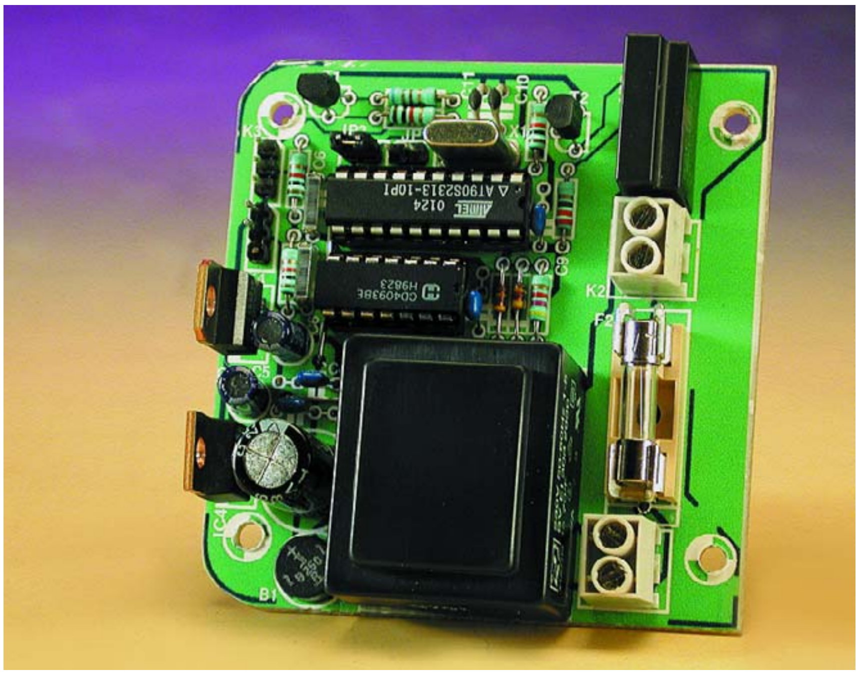
Maplin, in addition to a water cooling kit, has a TEC, and a range of fans, heatsinks and thermal compounds, for the home constructor.

Night Light Control

using the AT90S2313

Design by V. Schmidt

The special feature of this night light is that it has a switch-off delay. This is achieved using a microcontroller with an integrated 16 bit counter.



This night light can be switched on and off as normal, but it also includes a timer which can automatically turn the light off after a preset time. The light can be fitted with two (or more) buttons, one of which can be situated 'locally' to the light while the other (or others) can be some distance away. The local button can sport LEDs which not only make it easier to find it in the darkness, but which also indicate the status of the night light controller.

The circuit

Because the circuit uses a microcontroller, the number of components is kept down to a reasonable level. The circuit in **Figure 1** shows the AVR microcontroller type AT90S2313, which drives a solid state relay type S201S02 (IC5) from PD0 via T2. This relay is an electrically isolated switch connected in series with K1 (the mains supply) and K2 (the lamp). The relay includes a zero-crossing detector which turns on the connected load as the mains voltage goes through zero. The microcontroller and the rest of the circuit are powered from regulated DC supplies at +5 V (IC4) and +12 V (IC3), also, of course, isolated from the mains.

The two buttons are connected via K3 and K4 and drive port pin PB0 via a diode-OR configuration. In the quiescent state, when neither button is pressed, R2 and R3 hold the respective inputs of the inverting Schmitt triggers low, and so the outputs will be high. D4 and D5 allow the output level to be different from the input level, and R4 ensures that the microcontroller sees a high-level input voltage of 5 V. The gates are therefore also operating as level shifters. So why go to the trouble of using two different supply voltages? The reason is simple: with a possibly rather long cable between the controller circuit board and the buttons, a higher supply voltage greatly increases the noise immunity of the circuit at the cost of just one voltage regulator and three capacitors.

If one of the buttons is pressed, the level at the input to the corresponding gate goes high, and so its output goes low. Taking into account the forward voltage drop of the diode, the level at the input to the microcontroller will be around 0.9 V, which it will safely interpret as low. The signal is not debounced here: that job is done in software in the AVR microcontroller.

Transistor T1 is controlled via output PD1 or PD2, depending on which of jumpers JP1 and JP2 is fitted, and drives the LEDs next to button S1. If JP1 is fitted, the LEDs are normally illuminated continuously and flash when the sleep timer is running. If JP2 is fitted the LEDs are normally off but flash when the timer is running. The microcontroller is clocked at 4.9152 MHz. This unusual value is required to allow an internal signal at 10 Hz to be generated.

The software

So much for the hardware: now to the software. This was created and tested using the Atmel STK-500 starter kit. The WAVRASM assembler was used along with AVR Studio 3.2 for testing and downloading the code into the microcontroller.

The night light controller program is available for free download from the *Elektor Electronics* website in assembler and in compiled form. It is divided into six sections which we shall now discuss.

Initialisation (INI)

This is where the parameters are set that configure the functions of the internal components of the microcontroller. First the interrupt vector for Timer 1, running in compare mode, is set to the address of the interrupt code (label 'ini'). The compare interrupt for Timer 1 is now enabled by setting bit OCIE1A in register TIMSK. The prescaler for Timer 1 is set to divide by 1024 in register TCCR1B and the CTC option is enabled. When the preset value in registers OCR1AL and OCR1AH is reached, the timer is cleared and the process begins anew. This maximum value is now set: 30H in OCR1AL and 00H in OCR1AH. This corresponds to 48 in decimal. With the prescaler, and with a crystal frequency of 4.9152 MHz, the interrupt routine is called every 10 ms. Next the data direction registers for ports B and D are configured. Port B operates as an input, port D as an output. Finally the I bit is set in SREG to enable interrupts.

Main program

The main program consists of just four instructions, calling subroutines INPUT, LOGICOUT and OUTPUT. Then the main loop repeats from the beginning.

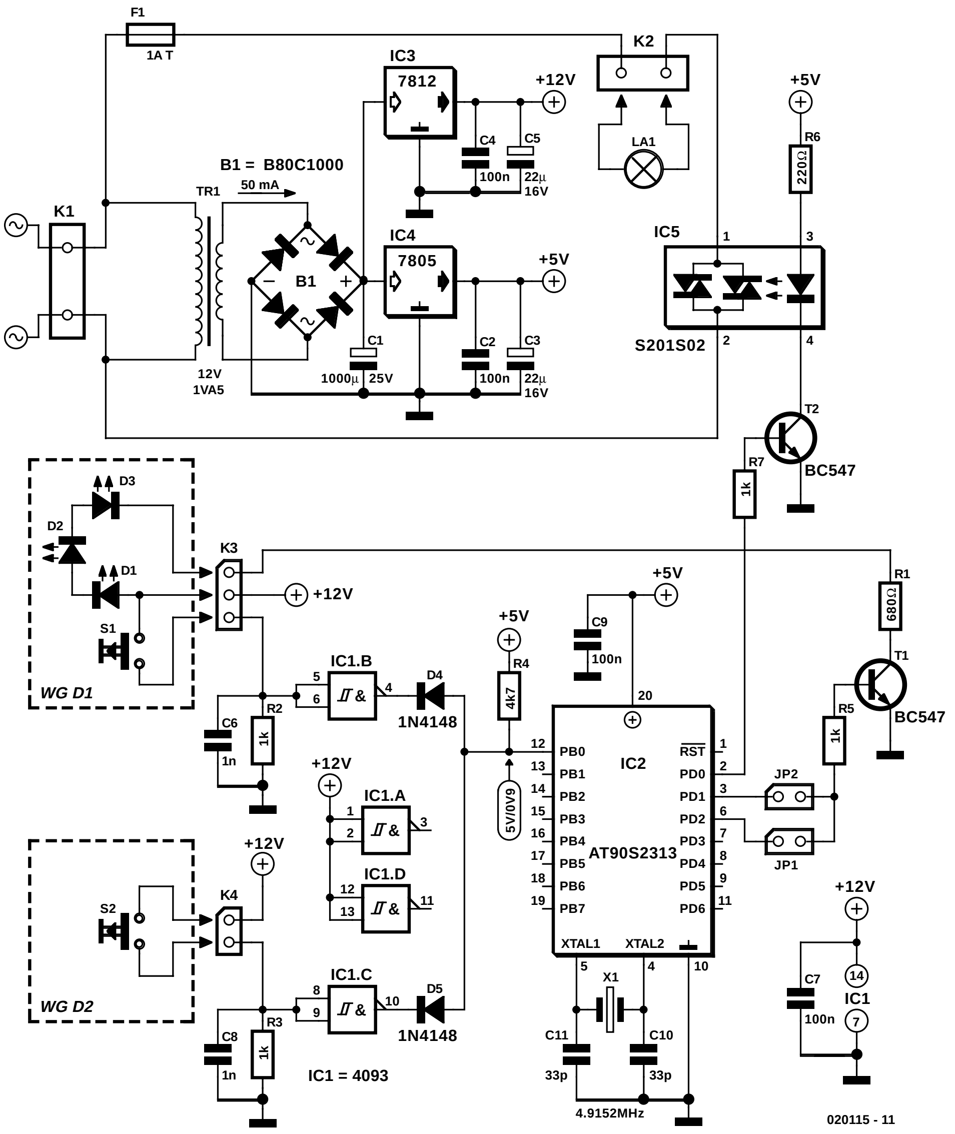


Figure 1. Circuit diagram of the night light controller with remote buttons.

Interrupt-driven timer routine (SUB TIMER)

The interrupt routine creates a basic clock period of 10 ms, which is used by DELAY for debouncing the buttons and incrementing register TAKT10 ('clock 10'), from which all time-dependent quantities are derived. TAKT ('clock'), which is used for measuring the duration of a button press, is incremented every 100 ms, when register TAKT10 reaches 10 (decimal). TAKT10 is then reset to zero and the process repeats from the beginning.

Next a one-second clock is generated for the 'status display' in subroutine OUTPUT, and a ten-second clock for the output timer (OUTC) in subroutine LOGICOUT. Flags F1SEC and F10SEC in STEUER ('control') are set according on the state of registers SEC1 and SEC10. Flag F1SEC is 1 for 500 ms and then 0 for a further 500 ms. Flag F10SEC is simply set every 10 s and subsequently cleared in subroutine LOGICOUT.

SUB INPUT

Subroutine INPUT is activated when port pin PB0 is taken low. When a button is pressed, the falling edge on the input starts DELAY for debouncing. On return from DELAY the port pin is tested again to verify that it is still zero. If this is the case the contents of register TAKT are copied into register START and flag FKENN ('edge detect') is set. Otherwise control returns to the main program. When the button is released the contents of register TAKT are copied into register STOP: this allows the period for which the button was pressed to be determined. Overflow is prevented by comparing registers START and STOP. If the period for which the button was pressed is less than 1.5 s (a count of 0F hex) flag FKL15 ('less than 15') is set; if it is greater, flag FGR15 is set. Then, in both cases, the edge detection flag FKENN is cleared.

SUB LOGICOUT

Subroutine LOGICOUT contains the logic for generating the output and controlling the sleep timer in register OUTC. First a check is made to see whether flag FKL15 or FGR15 is set. In the first case flag OUT in register STEUER is toggled (and the lamp will then be either turned on or turned off). In the second case flags OUT and OTIML are set. OTIML is the start condition for the sleep timer in reg-

AT90S2313 register settings

Timer 1 settings

Timer interrupt: TIMSK

TOIE1	OCIE1A				TICIE1		TOIE0	HEX
0	1	0	0	0	0	0	0	40

(Timer/Counter 1 compare match interrupt enabled. An interrupt is generated when the counter value becomes equal to the compare register contents.)

CTC and prescaler: TCCR1B

ICNC1	ICES1			CTC1	CS12	CS11	CS10	HEX
0	0	0	0	1	1	0	1	0D

(Clear Timer/Counter 1 to zero when count value equals compare register; set prescaler to divide by 1024 (CS12 to CS10).)

Compare register

OCR1AH	00h
OCR1AL	30h

0030H = 48 decimal

(Timer 1 operates as a clock generator and generates an interrupt every 10 ms. The crystal frequency of 4915200 Hz is divided down to 100 Hz by the prescaler and Timer 1.)

Stack pointer setting

SPL = DFh

Port control settings

Register DDRD	FFh: port D configured as output
Register DDRB	00h: port B configured as input

SREG setting

Set I bit to enable interrupts.

ister OUTC. This register is incremented every 10 s until it reaches the constant value OUTIM (180 decimal). Then both OUT and OTIML are cleared and the program returns to its initial state.

SUB OUTPUT

OUTPUT controls the output on port D. The action of this subroutine depends on the state of flag OUT. If it is set, it jumps to label OUTPUT1. There PD0 is switched high, and then the state of flag OUTIML is used to control further execution. If this flag is clear, subroutine OUTPUT terminates with a return to the main program. If, however, it is set, the succeeding assembler instructions ensure that 1 Hz clock signals are output on port pins PD1 and PD2, 180 ° out of phase with one another.

If flag OUT is not set port pins PD0 to PD2 are set to their default levels (PD0 and PD1 to zero, PD2 to one) and the subroutine ends.

SUB DELAY

Subroutine DELAY debounces the buttons using the 10 ms counter TAKT10. When the routine is called the counter's state is copied into register DELAY and the value incremented by 2, representing 20 ms. If this results in a value greater than ten, ten is subtracted from DELAY. This is important, since counter TAKT10 is reset to zero when it

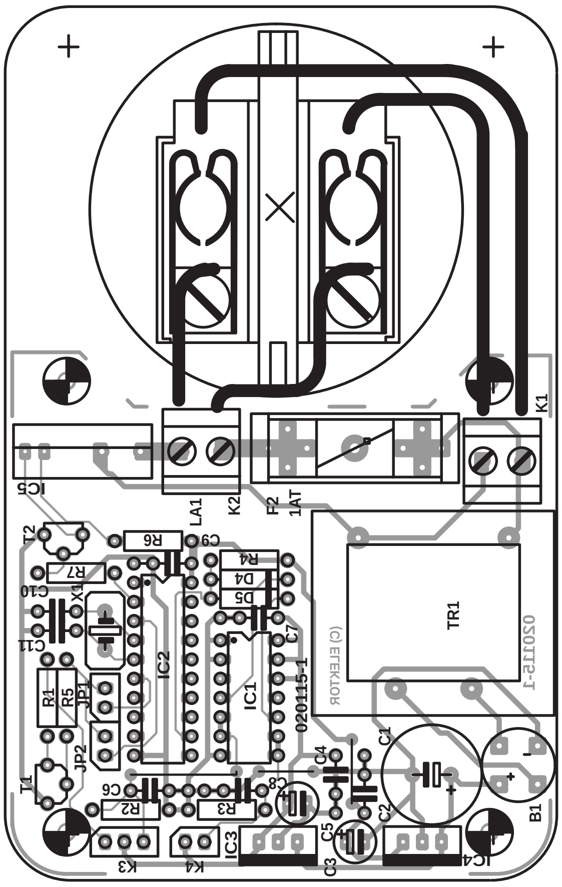
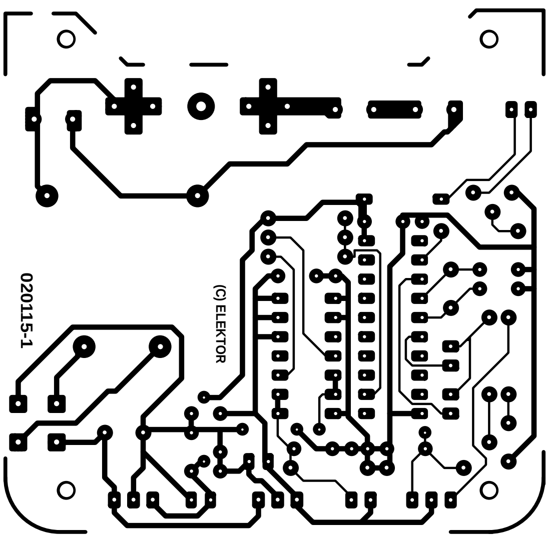


Figure 2. The night light controller is built in an enclosure with a mains plug and socket (board available from The PCBShop).



COMPONENTS LIST

Resistors:
R1 = 680Ω
R2,R3,R5,R7 = 1kΩ
R4 = 4kΩ7
R6 = 220Ω

Capacitors:
C1 = 1000µF 25V radial
C2,C4,C7,C9 = 100nF
C3,C5 = 22µF 16V radial
C6,C8 = 1nF
C10,C11 = 33pF

Semiconductors:
B1 = B80C1000 bridge rectifier in round case (80V piv, 1 A)
D1,D2,D3 = LED
D4,D5 = 1N4148
T1,T2 = BC547
IC1 = 4093
IC2 = AT90S2313-10PC (order code **020115-41**)
IC3 = 7812
IC4 = 7805
IC5 = S201S02 or S201SE2

Miscellaneous:
F1 = fuse, 1A(T) (time lag) with PCB mount holder
JP1,JP2 = 2-way jumper
K1,K2 = 2-way PCB terminal block, lead pitch 5mm
K3 = 3-way pinheader or solder pins
K4 = 2-way pinheader or solder pins
S1,S2 = pushbutton, 2 make contacts (see text)
TR1 = mains transformer, 12V 1.5VA (e.g, Hahn BV EI 302 2022)
X1 = 4.9152MHz quartz crystal
Disk, hex and source code, order code **020115-11** or free download

reaches ten. The following loop is then executed until the contents of DELAY and TAKT10 become equal. Then the subroutine returns.

Mechanical construction

The circuit is built inside an enclosure with a mains plug and socket. The tiny circuit board in **Figure 2** may have to be filed down slightly to fit inside the enclosure of your choice. There should be no difficulties with fitting the components and the circuit should work first time as long as the components are properly soldered and the three wire links are not overlooked.

The button (or buttons) can if desired be mounted separately in their own enclosures. The cable is taken through the underside of the enclosure (not forgetting a strain relief) and connected to terminals K3 and K4. It is sensible, however, to fit S1 along with its LEDs in the main enclosure.

(020115-1)

Free Downloads

– Night light controller program in assembler and in compiled form (source code and HEX file).
File number: **020115-11.zip**

– Printed circuit board layout in PDF format.
File number: **020115-1.zip**

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

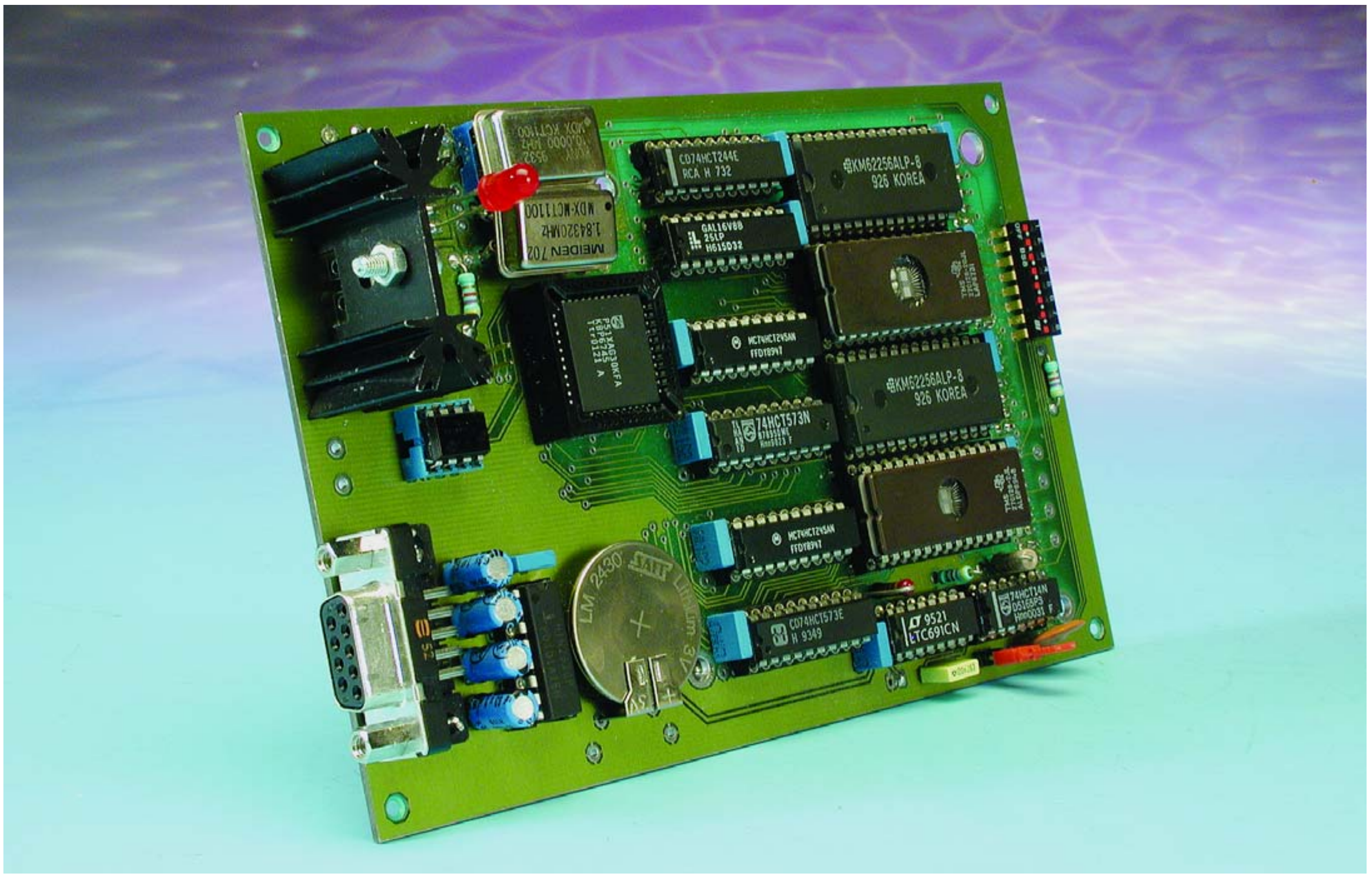
Universal XA Development Board

Part 2: construction and use

Design by B. Bouchez

Last month we examined the theoretical aspects of the universal development board for the Philips XA series of microcontrollers. In this second and concluding part, we describe the construction of the board and programming considerations.

Although the circuit is rather large and complex, the *Elektor Electronics* laboratories succeeded in fitting everything onto a single Eurocard (100 × 160 mm). The circuit board layout also gives proper consideration to the placement of he PC/104 connector and the associated fitting holes. Given the high component density, there was no alterna-



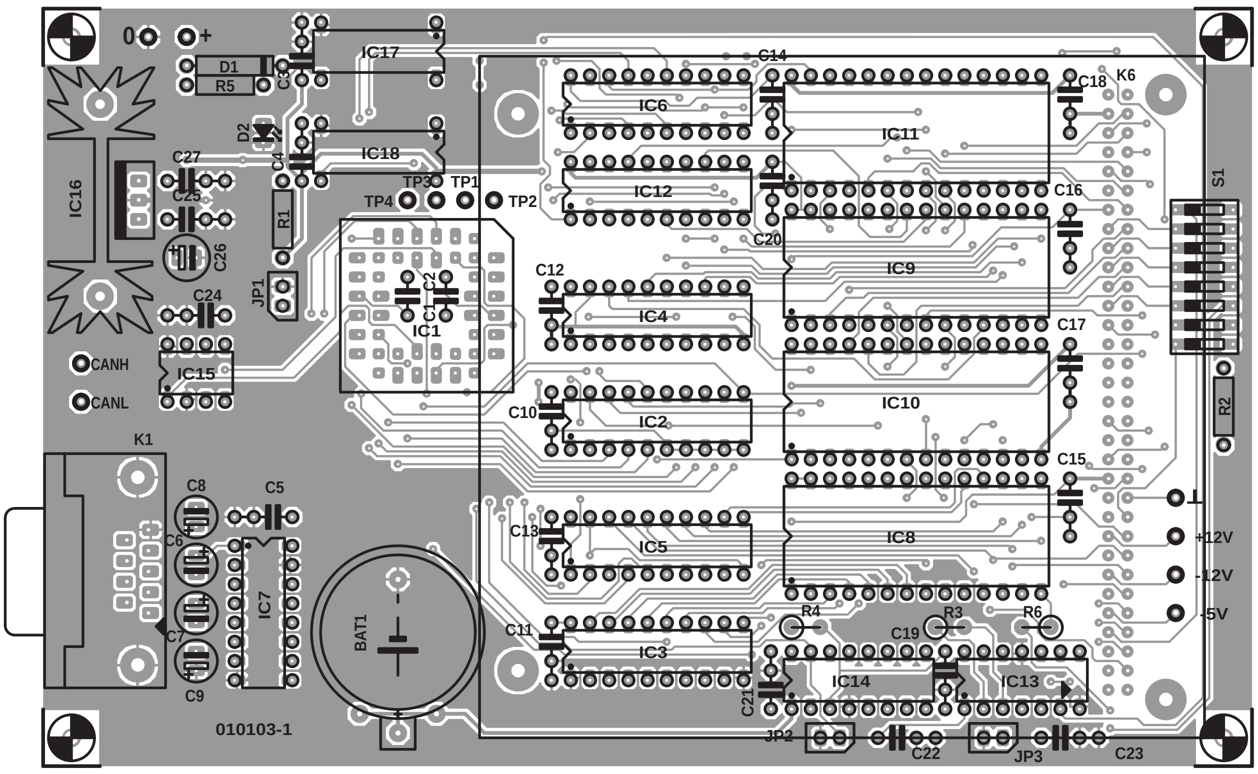


Figure 1a. Component layout for the XA development board.

COMPONENTS LIST

Resistors:
R1,R2,R3,R6 = 4kΩ7
R4 = 1kΩ5
R5 = 1kΩ

Capacitors:
C1-C5,C10-C21,C24 = 47nF
C6...C9 = 1µF 25V radial
C22,C27 = 100nF
C23 = 2nF2
C25 = 220nF
C26 = 10µF 35V radial

Semiconductors:
D1 = 1N4001
D2 = LED, low-current
IC1 = PXAC37KBA (Philips Semiconductors, see text)
IC2,IC3 = 74HCT573
IC4,IC5 = 74HCT245
IC6 = 74HCT244
IC7 = MAX232 (Maxim)
IC8,IC9 = 27C256-90, programmed, IC8: order code **010103-21**, IC9: order code **010103-22**)
IC10,IC11 = 62256–55 (RAM)
IC12 = GAL16V8 (programmed, order code **010103-31**)
IC13 = 74HCT14
IC14 = ADM691A
IC15 = PCA82C250T
IC16 = L7805CP
IC17 = 16MHz oscillator module
IC18 = 1.8432MHz oscillator module

Miscellaneous:
BAT1 = 3 V Lithium battery
JP1,JP2,JP3 = jumper
K1 = 9-way sub-D socket (female), PCB mount, angled pins
K6 = PC/104 connector, two rows of 32 contacts (see text)
S1 = 8-way DIP switch
Heatsink for C16
PCB, order code **010103-1**
Disk, contains source, hex and Jedec files, XADEV; order code **010103-11** or Free Download

tive to using a two-sided, through-plated circuit board layout.

Construction

The copper layout and component layout of the circuit board are shown in **Figure 1**. In general, fitting the components should not present any problems, but certain aspects — particularly fitting the PC/104 connector — require extra comment.

As usual, we highly recommend using sockets for the ICs, especially the memory chips. For the PLCC package of the XA microcontroller, there's no other choice, since it's hard to imagine how anyone could solder the 44 leads of this IC package (arranged in two closely spaced concentric squares) directly to the board. You should preferably use high-quality sockets with turned pins, in order to avoid problems over the long term.

The monitor EPROMs

IC8 and IC9, which are type 27C128 EPROMs, contain the monitor program that allows you to develop and test your own programs. They are available from Readers Services under order numbers **010103-21** and **010103-22** (these are type 27C256 devices, for reasons described further on). If you wish to program the EPROMs yourself, you can use the Intel hex files available from the *Elektor Electronics* website or on diskette from Readers Services (order number **010103-11**). There are two versions of these files. Files 010103-21 and 010103-22 are for use with 27C256 EPROMs, which can be fitted directly on the board in place of the prescribed 27C128s without any modifications. The hex files for 27C128 EPROMs are **010103-23** and **010103-24**. Be sure to use the correct files for whichever type of EPROM you have, and

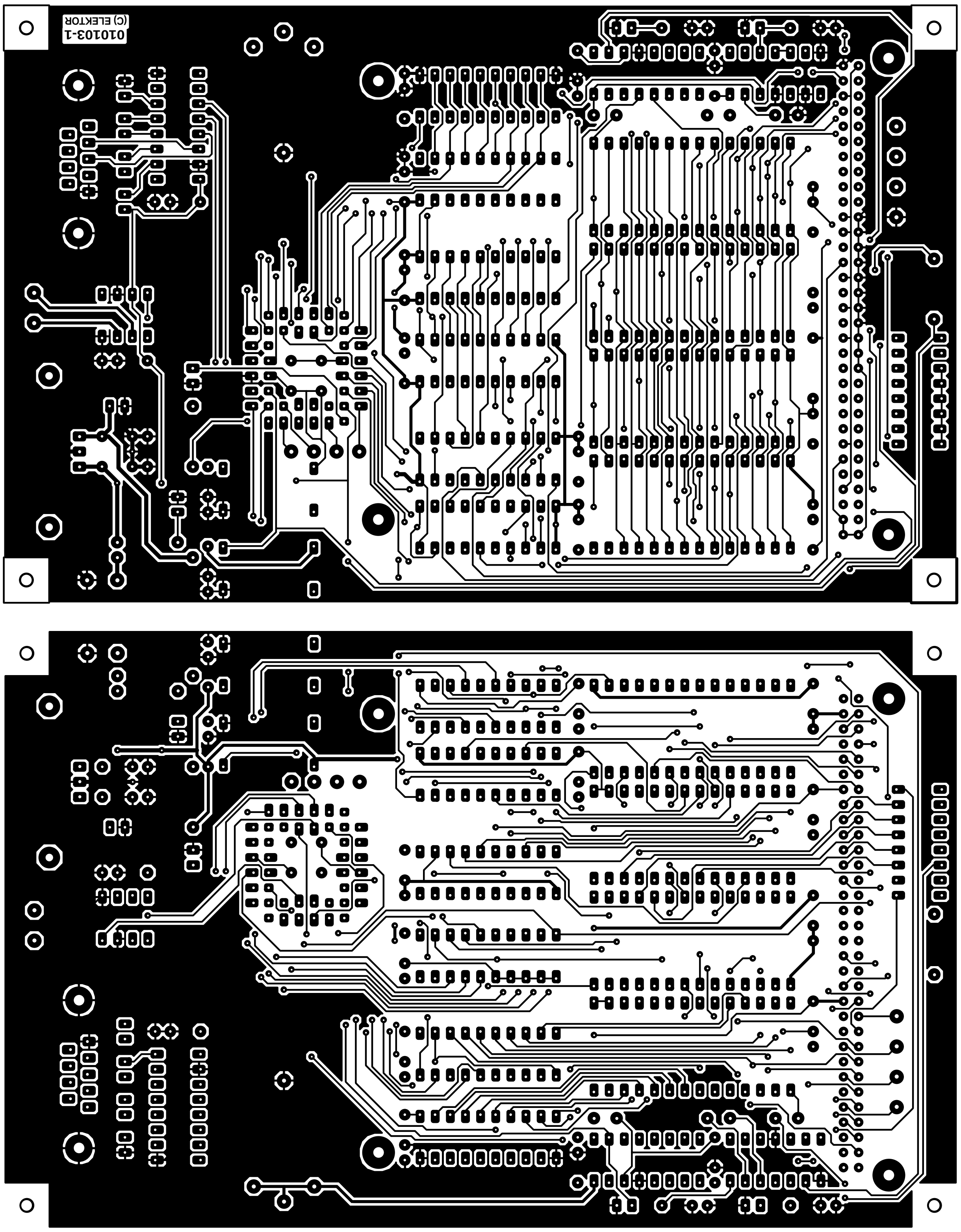


Figure 1b. PCB copper layout for the XA development board (double-sided, through-plated, available ready-made).

clearly label IC8 and IC9 so you don't accidentally fit them in the wrong sockets (swapped around).

The EPROMs must be CMOS types, since otherwise the circuit will not work. This means you must use either 27C128s or 27C256s, not 27128s or 27256s! In addition, all of the logic ICs must be 74HCT types. Do not mix ICs from different families on the board.

The board was originally designed to use 27C128 EPROMs, but it turns out that these devices are currently more difficult to obtain and more expensive than the larger-capacity 26C256. Fortunately, both types can be used without any modifications to the board. In the

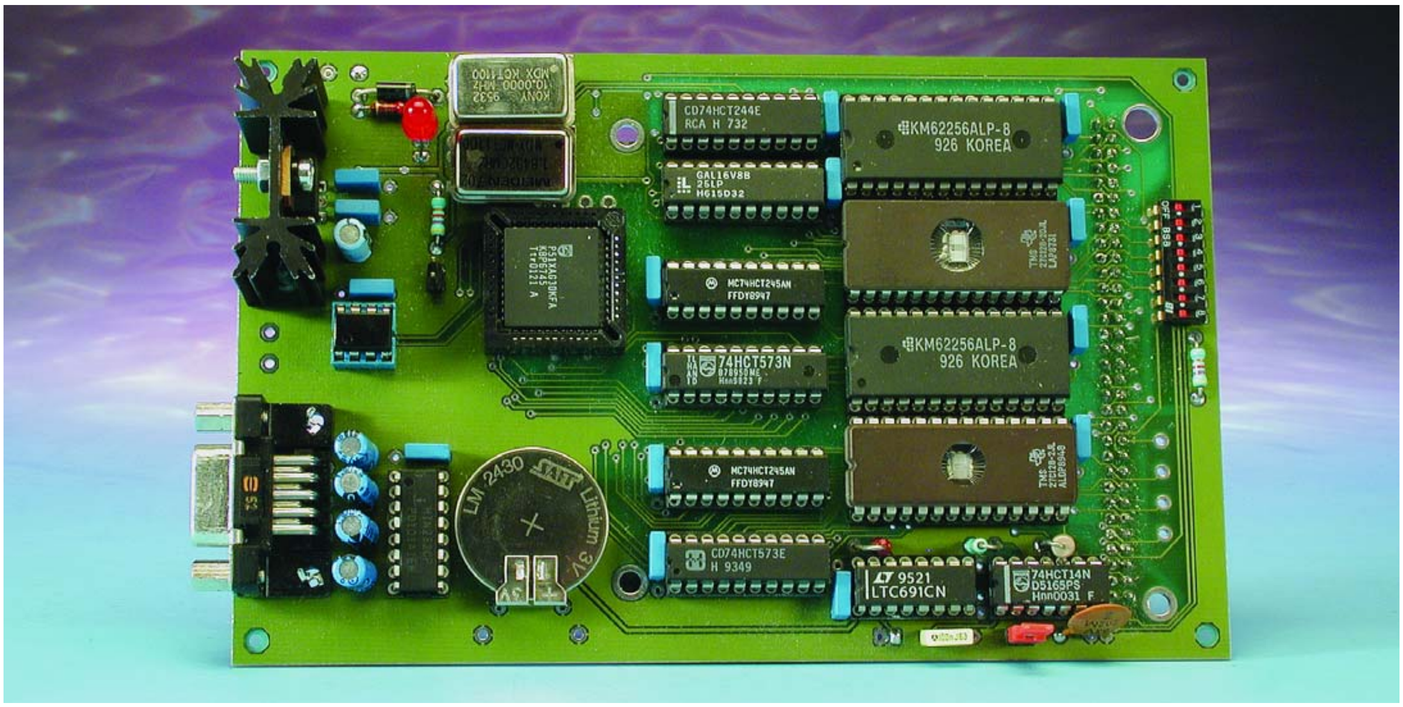


Figure 2. Photograph of a prototype of the board. In the final version, the oscillator packages are further apart.

27C256, the $\overline{PGM}$ (program) input of the 27C128 is replaced by the A14 address line, and on the development board this pin is tied to +5 V. The contents intended for the smaller 27C128s can thus be used with 27C256s by programming them to start at address $4000 instead of $0000. This is why two versions of the hex files are available.

Note: the current version of the monitor program is only suitable for use with XA-G3 microcontrollers. It is not compatible with the XA-C3. A special version for the latter type of microcontroller is currently under development, and we will advise you when it is available.

The GAL (IC12) is also available pre-programmed under order number **030103-31**. If you have the necessary equipment for programming GAL devices, you can also download the source file (GAL_IC12.SRC) and Jedec file (GAL_IC12.JED) from the *Elektor Electronics* website. If you do not have access to the Internet, these files are also available on the same diskette as the EPROM files (order number **010103-11**).

The monitor uses the slowest possible bus speed, which means that EPROMs with an access time of 250 ns are satisfactory. However, there is nothing to stop you from modifying the contents of the BTRH and BTRL registers in order to 'soup up' the circuit, as long as you use correspondingly fast memory ICs.

If you use an XA-C3 (with a corresponding monitor program), you must also fit IC15 (PCA82C250T), since it is necessary for access to the CAN bus. The circuit board is designed for a DIL version of this IC. If you can only obtain an SMD version, you will have to use an adapter socket.

PC/104

The PC/104 socket requires a bit of extra attention. There are two different types available: stackable and non-stackable.

The stackable type is actually a hermaphroditic connector, which means it has a male connector (header) on one side and a female connector (socket) on the other side. The socket is fitted on the component side of the circuit board, with the header passing through the board to the solder side. The pins of the header can then be inserted into the socket of a PC/104 card underneath the development board.

There is also a simpler, non-stackable type of connector consisting only of the header. This type of connector is most commonly found on controller boards (bus masters). The development board can be used with both types of connectors, as can be seen from lead photo for this article.

The standard height of a PC/104 socket is 0.6 inch (1.52 cm). If the ICs are fitted in sockets, they may be higher than this, which will make it impossible to fit PC/104 cards on top of the development board. In that case, the expansion boards can be connected to the bottom side of the board.

The stackable type of connector is often difficult to obtain. However, you can also use standard 2-row pin headers with a standard pitch of 2.54 mm and solder them to the bottom (copper side) of the circuit board. If you use a non-stackable connecter (socket), it must be fitted to the component side of the board.

Fitting the other components should not present any difficulties, so little additional description is necessary. All ICs have the same orientation. The space between the IC sockets is limited, so you should use thin decoupling capacitors, preferably Siemens Sibatit (miniature ceramic) types. Capacitors C1 and C2 are fitted to the solder side of the board, as shown in **Figure 3**.

Using the development system

Up to now, we have only described the hardware of the development system. However, a microcontroller board without any software is totally useless. Besides this, you will most

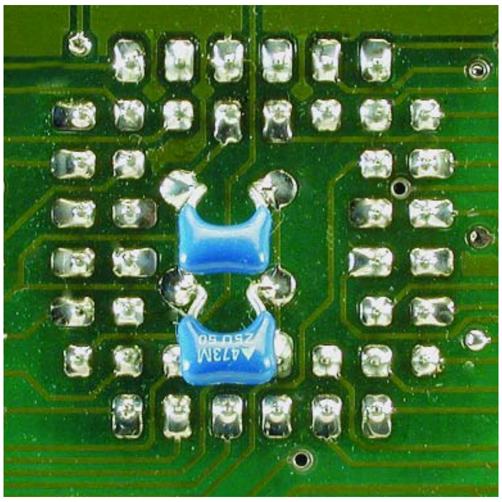


Figure 3. Close-up of the placement of C1 and C2 at the solder side of the board.

likely have to pass through a learning stage before you will be able to write your own stand-alone programs. In order to make this unavoidable process easier, we have written a simple but powerful monitor program that you can use to test your own programs under real conditions.

A simple terminal emulator program configured for 9600 baud, 8 data bits, no parity, 1 stop bit and no flow control is all that is necessary for communication with the monitor program. Programs such as HyperTerminal and Procomm are excellent choices for this task. The link between the PC and the development board is provided by a standard 1:1 RS232 cable (**not** a null-modem cable!).

After giving the assembled board a final inspection, you can connect it to a free COM port on your PC, start up the terminal emulator and switch on the power. The following message should appear on the screen of the PC:

***** XA-G3 Monitor V1.03 *****

(followed by some additional text), after which a command prompt (>) should be displayed, as shown in the screen dump in **Figure 4**. This indicates that the microcontroller board is waiting for further commands.

The monitor program uses relatively few commands, and experienced users of development systems should find them fairly familiar. **Table 1** presents a summary of the monitor commands.

Programming in XA assembler

It's nice to have a flexible, high-performance development board, but that's the easy part. The hard part is using it. One of the most difficult aspects of working with a new microcontroller, for hobbyists as well as for professionals, is acquiring the necessary hardware and software.

As far as the hardware is concerned, this development board provides a large number of features at a reasonable price. To achieve even more flexibility, it would be necessary to use an in-circuit emulator (ICE), which is a significantly more expensive proposition.

The software (assembler, compiler, line editor, library manager and simulator) can also easily cost a pretty penny. However, the software for this system is free, at least as long as you program in assembler, since Philips Semiconductors offer a set of semi-professional development tools for the unbeatable price of 0 (say, zero) pounds. Naturally, if you wish you can later acquire more extensive development tools, such a relative assembler and/or C compiler.

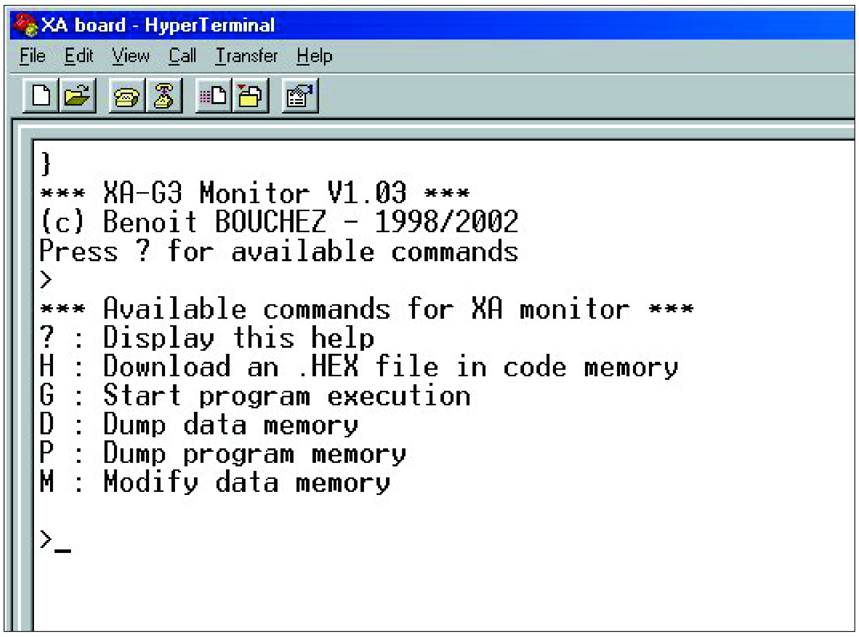
The Philips Semiconductors development package, which is called XADEV, contains an absolute assembler, a simulator and a development environment, all of excellent quality. This software is no longer supported by Philips Semiconductors and is written for use with Windows 3.1, but it still works perfectly with more modern 32-bit versions of the operating system (at least Windows 95 and Windows 98; we haven't tested it with Windows NT4, 2000 or XP).

This development package is available on diskette and from our website (with the permission of Philips Semiconductors). The software is compressed into a zip file (**XADEV.zip**), so it must first be unpacked on a diskette or in a subdirectory on the hard disk before it can be installed. The only negative point to note about this software is that it is missing a user manual. Although the Help function provides a certain amount of assistance, it is not very extensive. Fortunately, we were able to find a manual that was written by a Philips Semiconductors application engineer. According to the author, a scanned version of this manual is available for download from his website at

http://benoit.bouchez.free.fr/

The file to look for is called **XADEV_manual.zip**.

As can be seen from the screen dump in **Figure 5**, this development environment, despite its age, can hold its own with more recent models. After the source code has been written using the editor, a simple click on a button in the menu bar is all it takes to assemble the code. The application can then be run in the simulator in the 'run' or 'single-step'



```
}
*** XA-G3 Monitor V1.03 ***
(c) Benoit BOUCHEZ - 1998/2002
Press ? for available commands
>
*** Available commands for XA monitor ***
? : Display this help
H : Download an .HEX file in code memory
G : Start program execution
D : Dump data memory
P : Dump program memory
M : Modify data memory

>_
```

Figure 4. The greeting message of the monitor program as received by HyperTerminal.

modes, variables and registers can be modified as desired, and so on. In short, there's no reason for complaint!

A comment is in order here regarding the data memory of the XA family. Unlike the 8-bit processors of the 8051 family, no distinction is made between internal and external RAM. In the XA family, the data memory is a single large block ranging from address $0000 to address $FFFF. The internal memory always starts at address $0000, and depending on the type of processor, it is either 512 or 1024 bytes. The external memory is automatically selected if the memory location being addressed lies outside the range of the internal RAM. There is also another important difference, which is that registers R0–R15 are located in a separate, dedicated memory region, instead of in the regular RAM.

Another consideration is access to the special function registers (SFRs). The contents of these registers cannot be read or written using the monitor program, since they are addressed in a special manner that is not compatible with the indexed addressing method used by the D and M commands of the monitor program. If you need to debug a program at the register level, you will have to specifically insert MOV instructions in the program for the registers in question (which is not difficult).

After you have tested your program using the simulator, the hex file generated by the assembler can be loaded into the program memory of the development board. As already mentioned, this can be done using a standard terminal emulator program, such as HyperTerminal or Procomm.

Before you start programming, we emphatically suggest that you download some sample programs from the Philips Semiconductors website listed under 'Internet addresses' and carefully study the source code. A good starting point is the SKEL.ASM program, which forms part of the XADEV package. This program contains the minimum outline of an assembly-language program for an XA microcontroller.

Bear in mind that the XA family has very little in common with its 8-bit relatives and demands a completely different manner of programming, which is bound to confront you with a number of surprises!

(010103-2)

Table 1. Monitor program commands

Command	Function
?	Display the available commands and brief descriptions.
P	Display the contents of the program memory.
D	Display the contents of the data memory.
M	Modify one or more bytes.
G	Run the application ('go').
H	Download a hex file.

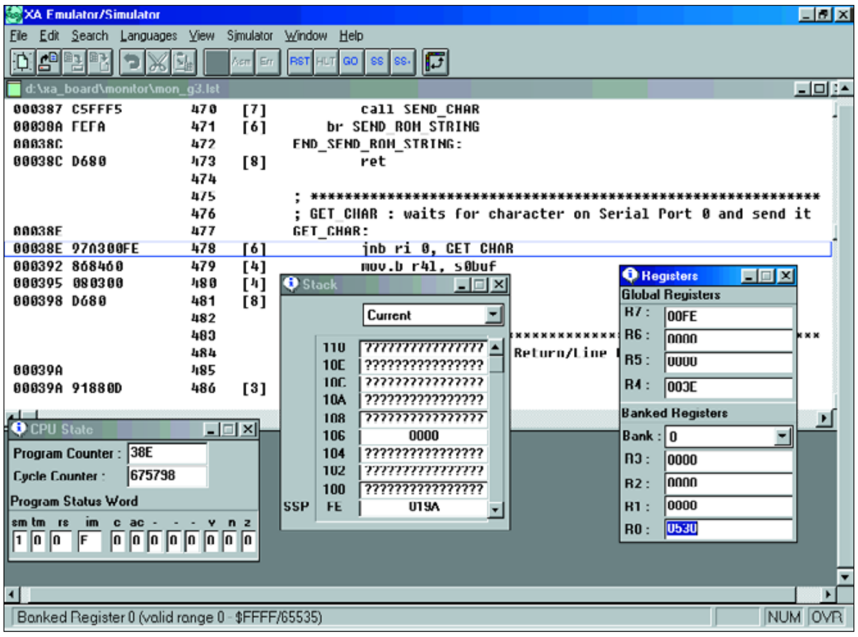
Internet addresses

Author's website:
http://benoit.bouchez.free.fr

Philips Semiconductors home page:
www.philips-semiconductors.com

Philips sample XA assembler programs:
www.semiconductors.philips.com/markets/mms/products/microcontrollers/support/software_download/16bit_xa/index.html

Raisonance
(demo version for the XA family):
www.raisonance.com/download/index.php


XA Emulator/Simulator
File Edit Search Languages View Simulator Window Help
Asm Err RST HLT GO SS SS.
d:\xa_board\monitor\mon_g3.lst
Stack
Registers
CPU State

```
000387 C5FFF5         470   [7]           call SEND_CHAR
00038A FEFA           471   [6]       br SEND_ROM_STRING
00038C                472         END_SEND_ROM_STRING:
00038C D680           473   [8]           ret
                      474
                      475         ; ***************************************************************
                      476         ; GET_CHAR : waits for character on Serial Port 0 and send it
00038E                477         GET_CHAR:
00038E 970300FE       478   [6]           jnb ri 0, GET CHAR
000392 868460         479   [4]           mov.b r4l, s0buf
000395 080300         480   [4]
000398 D680           481   [8]
                      482
                      483
                      484
00039A                485
00039A 91880D         486   [3]
```

Stack: Current

```
    110  ????????????????
    10E  ????????????????
    10C  ????????????????
    10A  ????????????????
    108  ????????????????
    106        0000
    104  ????????????????
    102  ????????????????
    100  ????????????????
SSP  FE        019A
```

Registers — Global Registers: R7: 00FE, R6: 0000, R5: 0000, R4: 003E
Banked Registers — Bank: 0, R3: 0000, R2: 0000, R1: 0000, R0: 0530

CPU State — Program Counter: 38E, Cycle Counter: 675798

Program Status Word

sm	tm	rs	im	c	ac	-	-	-	v	n	z
1	0	0	F	0	0	0	0	0	0	0	0

Banked Register 0 (valid range 0 - $FFFF/65535) NUM OVR

Figure 5. Screen dump of the XADEV development environment.

Table 2.

Contents of diskette 010103-11

GAL_IC12.SRC	Source file for the GAL
GAL_IC12.JED	Jedec file for the GAL
Mon_g3.asm	Source file for the monitor program
Mon_g3.hex	Intel HEX file for the monitor program
010103-21.hex	Content of EPROM IC8 (for 27C256)
010103-22.hex	Content of EPROM IC9 (for 27C256)
010103-23.hex	Content of EPROM IC8 (for 27C128)
010103-24.hex	Content of EPROM IC9 (for 27C128)
XADEV.zip	Compressed XADEV development environment package

CompactFlash (CF) Interface

For MCS-BASIC52 systems

Design by G. Meers

Prices of CompactFlash cards having dropped considerably over the past year or so, the author decided to design a parallel PC interface that allows data or programs to be written into one of these handy solid-state memories using MCS-BASIC52.

CompactFlash (CF) cards are gaining popularity every day, probably because the vast majority of digital cameras employ them for mass data storage, and, in a more general sense, because the phenomenon 'digital camera' itself is taking the world by storm. Thanks to ongoing developments in CF technology, the memory capacity rises while prices drop. This

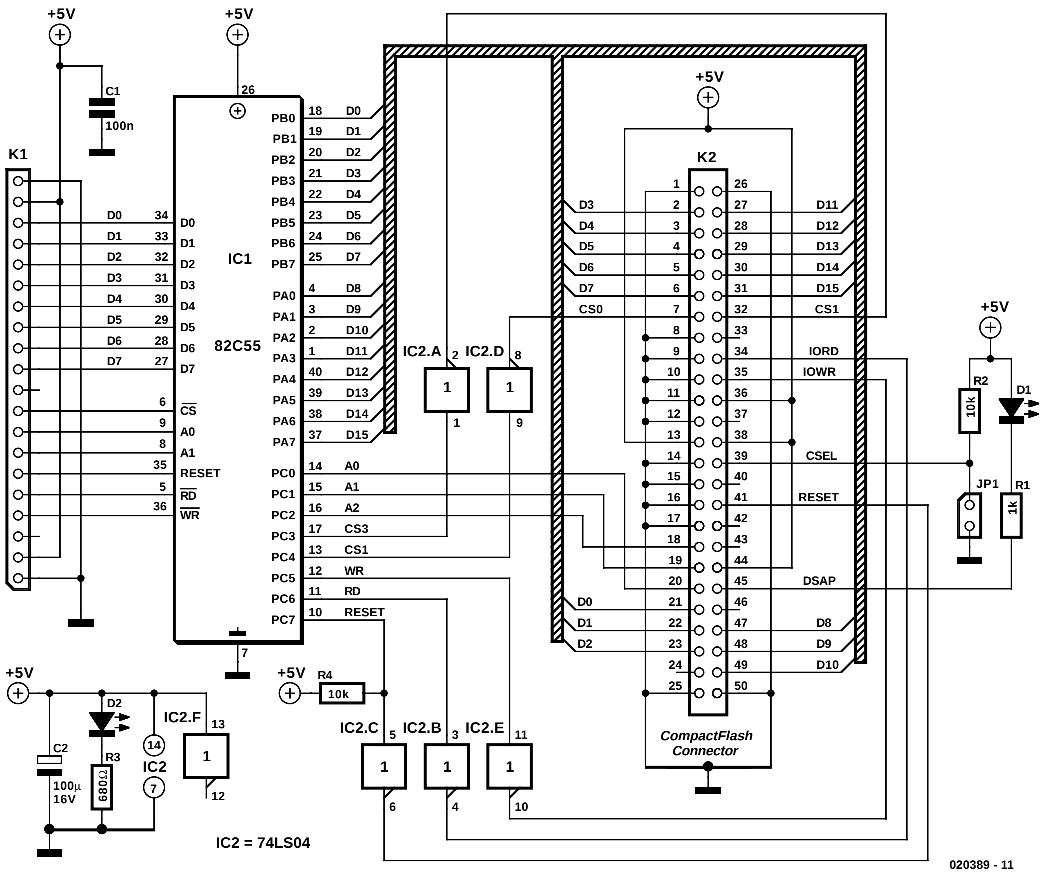


Figure 1. The required hardware is down to a minimum.

makes the CompactFlash card particularly attractive for PC users too, and you should not have been surprised by us publishing a CompactFlash-to-IDE Interface as early as April 2002.

Triggered by the above mentioned publication, the author set out to design a CF interface for connection to the address bus, data bus and control bus of the PC. The resultant circuit, in combination with some simple software, allows BASIC52 programs to be stored on a CompactFlash card.

The software developed for this project may be downloaded free of charge from our website, www.elektor-electronics.co.uk as file number **020389-11.zip** which may be found under month of publication. The program reports attempts to enter incorrect parameters or undesirable write actions to memory blocks. The interface draws a current of about 42 mA and achieves a data throughput rate of about 50 kB per second. The author actually employs the interface on a 20-MHz system with excellent results and reliable data transfers.

Circuit diagram

The circuit diagram of the CF Interface is shown in **Figure 1**. As you can see, the required hardware remains limited to an 82C55 chip, a 74LS04, two indicator LEDs and two connectors. With some luck, many of the components may be found in the junkbox.

A 50-way connector, K2, allows the CF card to be inserted. On this connector, pin 9 is wired to ground, forcing the card to behave like an IDE hard disk. In this mode of operation, jumper JP1 (between pin 39 and ground) **must** be fitted.

The complete control of the IDE interface is handled by the 82C55 (IC1). Using its bi-directional ports PB0-PB7, the chip drives data lines D0-D7, while port lines PA0-PA7 handle data lines D8-D15. Port lines PC0-PC7 are programmed to look after the control lines that form part of the IDE interface. Using software, bi-directional ports PA0-PA7 and PB0-PB7 are set up as inputs or outputs at the proper instant. By contrast, port lines PC0-PC7 are always configured as outputs.

The inverters contained in IC2 are needed because the 82C55 pulls the relevant port lines Low when the data direction is reversed. When this takes place, the inverters prevent contention and undefined levels on the control lines.

The following signals have to be connected to the system by way of connector K1: 5 V and ground; D0-D8; RD; WR; A0, A1 and RESET; CS.

Note that the address decoder for CS (Chip Select) is **not** included in the circuit. The CS signal has to go Low when addresses from 0C000h are selected by software. The 82C55 requires four addresses:

DATA register PA, base address +0;
DATA register PB, base address +1;
DATA register PC, base address +2;
CONTROL register, base address +3.

A few more circuit details: LED D1 lights to indicate activity on the CompactFlash card. D2 indicates the presence of the supply voltage. Finally, capacitors C1 and C2 are included to decouple (filter) the supply voltage.

Software

The software for the interface was written in ASM51 and the source code may be found under two names:

- PARCFFU.ASM (for PCFF commands from MCS-BASIC52);
- PARCFF.ASM (for CALL commands from MCA-BASIC52).

The software comprises a rudimentary IDE driver that divides the CompactFlash card into blocks of 8,192 bytes. A cheap 64 MB CF card has a storage capacity of 7,828 blocks, or 64,086,016 bytes. That's slightly less than the nominal capacity of the card because the last virtual cylinder is not completely used in the software.

In the program source code, the 82C55 has its addresses assigned as follows:

```
P8255A    EQU    0C008H    ; PORT A
                             ADDRESS
P8255B    EQU    0C009H    ; PORT B
                             ADDRESS
P8255C    EQU    0C00AH    ; PORT C
                             ADDRESS
P8255CO   EQU    0C00BH    ; CONTROL
                             ADDRESS
```

If the CS signal on connector K1 selects other addresses, then the above allocations should be changed accordingly.

Because the software is intended as an extension of MCS-BASIC52 the system must provide for sufficient extra program memory starting at address 2000h. The program proper has a size of 1.6 kB. In case you've already extended the system with additional MCS-BASIC52 commands, obviously the source code has to be adapted and re-assembled. When doing so, bear in mind that the source code employs ACALL instructions with an addressing limit of 2 kB!

Finally, the software for the CF Interface is compatible with MCS-BASIC52 V1.1 and V1.3.

PARCFFU.ASM

This is the source code of the program you'll be using to extend MCS-BASIC52 with extra statements. The assembled (object) code called PARCFFU.HEX has to be copied into the system memory starting at location 2000h.

The following commands are then available:

- **PCFF V** – Show version of connected CompactFlash card.
- **PCFF R** – Reset CompactFlash card.
- **PCFF L, block number, address** – Load data from the CompactFlash card into the system. The 'block number' is the location on the card (0-7823) and has a fixed size of 8 kB. The parameter 'address' contains the start address of the system data memory (00000h through 0FFFFh). After each processed sector, an 'L' is sent to the terminal.
- **PCFF S, block number, address** – Save data from the system data memory into the CompactFlash card. As above, only an 'S' is sent after each sector.
- **PCFF F, block number** –This causes the complete contents of this block number on the card to be filled with the value 0FFh. An 'F' is sent on completion of a sector.
- **PCFF D, block number** – Show status of block number: B = data; V = empty.

Unambiguous error reports will pop up if the card is either not ready, the block number is too high or if the block is in danger of being overwritten.

Using the PCFF instructions

Programs developed under MCS-BASIC are contained in the RAM memory from address 0200h onwards. The instruction PCFF S, 10, 0200H writes the program to block 10 on the card. Using PCFF L, 10, 0200H the program may be placed into RAM memory again.

If the program is larger than 8 kB (find out using P. LEN) it has to be accommodated in multiple blocks. If this is required, you should note the following. The instruction PCFF S, 10, 0200H writes data from address range 0200H through 21FDH into block 10. This represents a total of 8,190 bytes, not 8,192 as you may have expected because two bytes are used to flag that the block already contains data. For example, to write (save) a program of 10 kB, the following instructions should be used:

```
PCFF S, 10, 0200H : PCFF S, 11, 21FEH
```

or

```
PCFF S, 10, 0200H : PCFF S, 11, 0200H+8190
```

Reloading the program is then done as follows:

```
PCFF L, 10, 0200H : PCFF L, 11, 0200H+8190
```

Variables may be used for the block number and the address. If you want to erase, say, block 100 to 120, the following little program may be run:

```
100     FOR T=100 to 120
101     PCFF F, T
102     NEXT T
103     END
```

PARCFF.ASM

In some cases, it is not possible to extend the system with extra MCS-BASIC52 statements starting at program address 2000H. That's why the source code contained in PARCFF.ASM is intended to enable the program to be fitted at any location in the system's program memory. In this source code, the start address is 0B000h and this may be changed to your requirements. The start address is defined using the following code chunk:

```
START EQU 0B000H
ORG   START
```

This allocation may be changed and a new program address created. Obviously, the source code then has to be assembled again.

The operation and practical use is identical with PARCFFU.ASM described above, only the letters 'PCFF' in the instructions have to be replaced by 'CALL 0B000H'. Information on the way the necessary variables are entered in assembly code may be found in the source code file.

(020389-1)

Free Download

– ASM51 source code and HEX files of PARCFFU and PARCFF. File number: **020389-11.zip**.

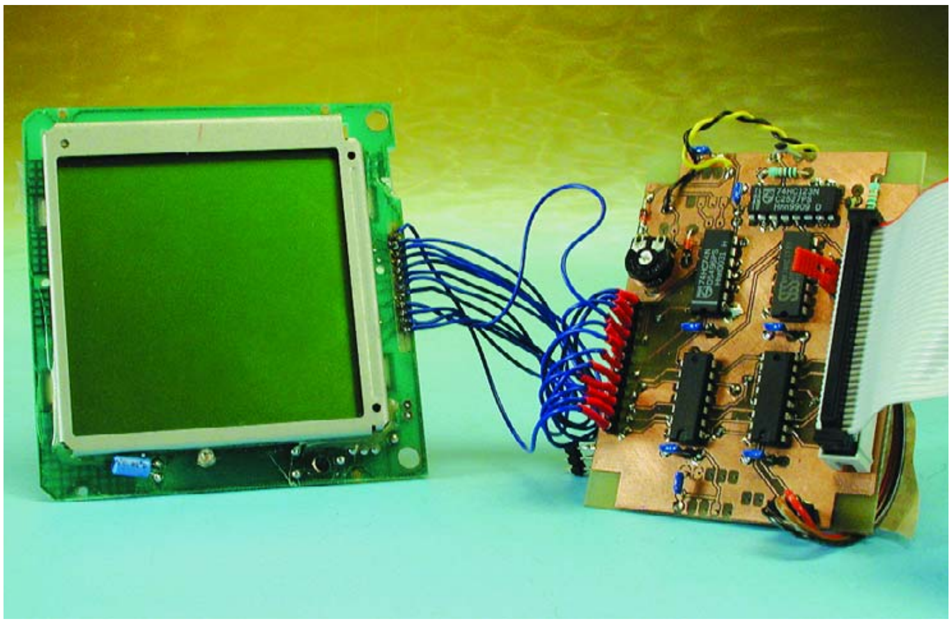
www.elektor-electronics.co.uk/dldl.htm
(select month of publication).

Low-Cost LCD Controller (2)

Part 2: the hardware

By Wim Huiskamp huiskamp@fel.tno.nl

In this month's instalment we apply the theoretical knowledge obtained from part 1 to make an 8051 microcontroller drive a matrix display salvaged from a computer game. Because the LCD requires relatively high data speeds, a hardware trick is used to keep the controller software overhead within reason.



To prevent flicker, matrix LCDs must be refreshed at frame rates greater than 50 Hz. The concept of the low-cost LCD controller is based on an 8051 microcontroller that creates and keeps an image in its external RAM and then reads the image at the required frame rate. The image databytes read from the external RAM are sent to the LCD display drivers. Clearly, performance optimisation is a major consideration if we are to achieve the required refresh rate and still preserve controller time for other tasks. So, instead of using a separate 'write' operation to send the RAM data from the CPU to the LCD drivers, a trick is applied in this design that allows the LCD to 'sniff' the image data while it is being read from the external RAM (see **Figure 1**). The data lines are effectively shared between the processor, RAM and LCD. The XSCL control for the LCD module is derived from the processor's $\overline{RD}$ control, and XSCL will clock-in the image RAM data at the right instant. Note that the actual image data read by the processor during refresh cycles are ignored, it being a 'dummy' read operation for the CPU. The solution reduces the CPU overhead for the image refresh operation by 50-60%. All remaining control signals for the LCD module (i.e., LP and FLM) are simply generated by bit-banging some of the CPU's port pins. These signals change at a much lower frequency than the data and XSCL and thus do not place a too heavy load on CPU performance. The FR signal is derived from FLM.

As a bonus to the performance advantage, several LCD controller functions come for free by using the already available CPU as the centre of the design:

– The CPU can access and update the display RAM as regular external RAM without additional hard-

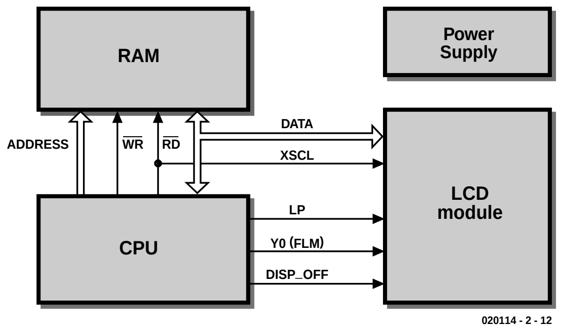


Figure 1. Block diagram of the Low-Cost LCD Controller.

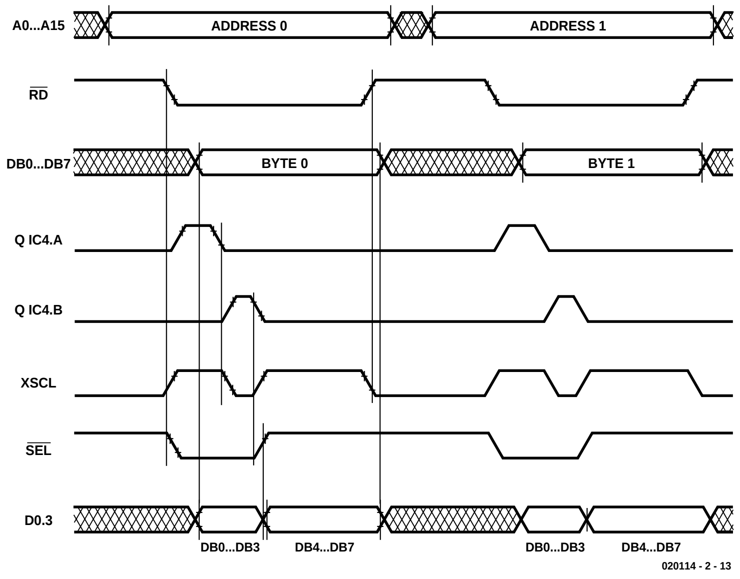


Figure 2. Timing diagram for the Low-Cost LCD controller.

ware. No additional image RAM is needed. Memory access contention between the CPU and the display refresh hardware is avoided.

- The normal CPU address bus hardware is also used during the image refresh cycles. Scrolling and panning of the LCD image is possible by selecting which area of the image RAM is 'read' during image refresh.
- No additional timing hardware is needed. The refresh cycle is driven by internal CPU Timer interrupts.

The CPU shares its data lines with the external RAM and the display. Therefore, some LCD control lines should only be enabled during the actual refresh cycles to avoid spurious image data appearing on the LCD during 'normal' CPU read operations on the memory. Some precautions are also necessary to disable the LCD while the CPU is not yet initialised (e.g., during and immediately after reset). Disabling the display during these phases is necessary to avoid damage to the LCD as a result of 'DC' signals on the matrix electrodes.

Details of the design and the way in which enable signals are generated will be discussed in the next section.

CPU interface and LCD controller

The LCD controller design assumes the use of a standard 8051 configuration with external RAM (e.g., 32k × 8) and the availability of the RD line and three control bits on Port 1.

The circuit has been tested with the AT89S8252 Flash Micro Board December 2001) and the High-Speed Controller Board (May & September 2002).

Figure 2 shows the relevant timing of the CPU refresh operation and the LCD control and data signals generated by the controller. The circuit diagram is given in **Figure 3**. A brief description of the design is given below.

The top traces of **Figure 2** show the CPU generating the Address lines A0-A15 and activates the RD control during the LCD refresh cycles. This results in the corresponding RAM data appearing on the databus AD0-AD7. The CPU shares the databus with the RAM and LCD controller. The datalines are connected to IC1, a quad 2-input multiplexer device. The four outputs of the multiplexer are connected to D3-D0 of the LCD module and clocked-in by the LCD on the falling edge of XSCL. This multiplexer device allows us to use all 8 bits of RAM data for the LCD, first nibble AD0-AD3, then nibble AD4-AD7. The nibble selection SEL is toggled automatically on each rising edge of XSCL by IC2A, a 74HC74 flip-flop. Mapping of the databits onto the LCD is arranged such that databit 0 appears leftmost on the display and databit 7 appears rightmost. This mapping can be modified by simple re-wiring of the multiplexer and D3-D0. IC2A is re-initialised to the correct nibble selection at each LP through inverter U3A.

XSCL is derived from RD. IC4c suppresses XSCL during normal RD operations of the CPU. This is achieved using the LP signal, which is kept logic High during normal CPU operation and thus disables XSCL. During refresh cycles the LP signal is pulled low and any RD activity during the refresh period will result in an XSCL. Sure, XSCL could have been derived directly from RD, but this would require two CPU read operations on the same address to allow the use of all 8 databits. Again, some simple additional hardware was added to reduce the CPU refresh load significantly. Monostable pulse generator IC5A (74HC123) is triggered by the RD signal, the resulting pulse at output Q of IC5A triggers IC5B (74HC123) and the resulting pulse at Q of IC5B effectively creates two XSCL cycles for each single RD operation. Obviously, the width of the pulse generated by IC5A (100 ns) must not violate the RAM access time and the combined pulse widths (IC5A, 100ns + IC5B, 50 ns) must be less than the RD cycle duration (about 500ns for a CPU clock of 11.0592 MHz). The XSCL doubling function may not be practical or desired when using a really fast version of the 80C51 CPU (e.g. the Dallas '420). Jumper JP1 may therefore be used to disable/enable the function.

IC3A and IC3B buffer the LP signal generated by the CPU. As explained before, LP is connected to IC45C to avoid spurious image data appearing during normal CPU opera-

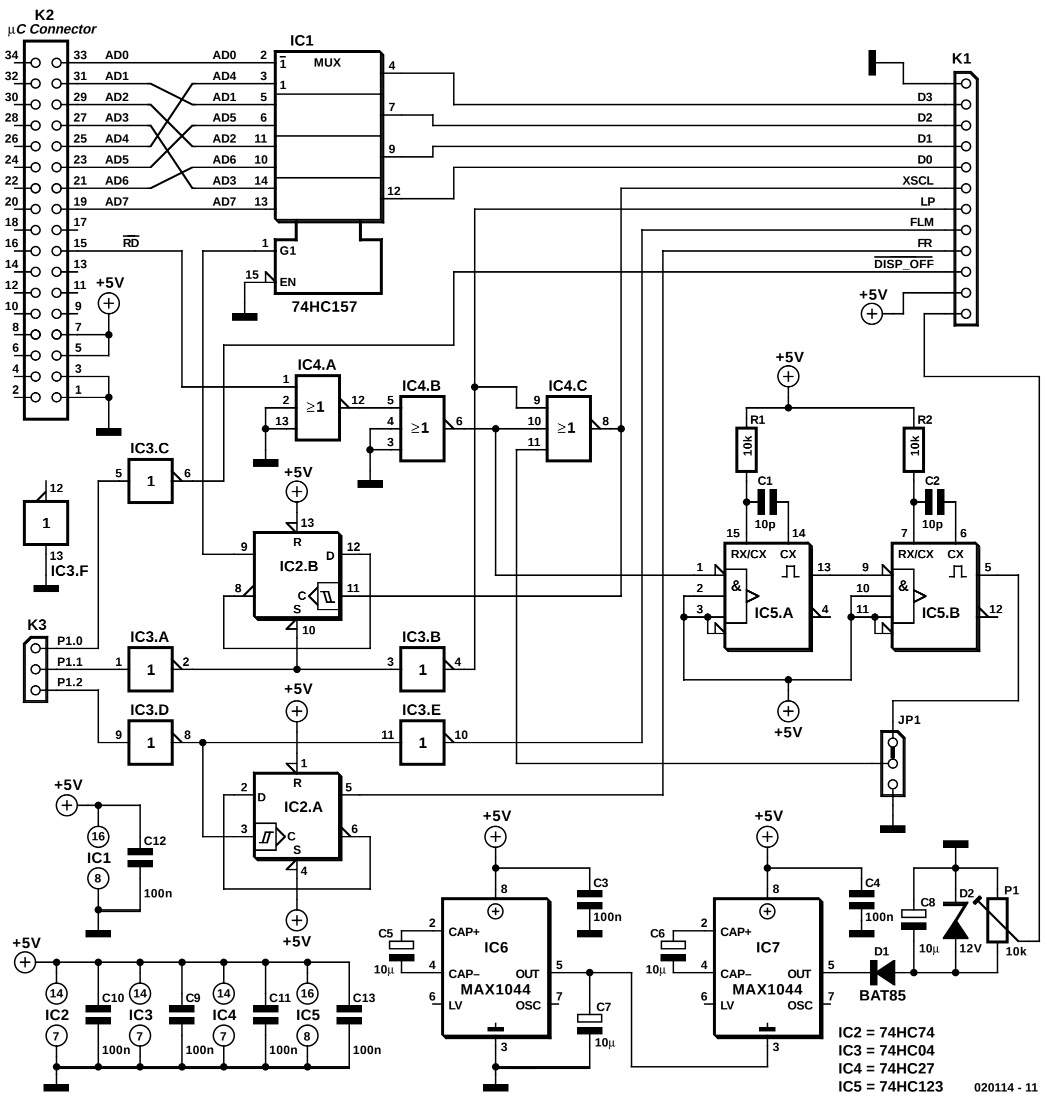


Figure 3. Circuit diagram of the LCD controller.

tion and LP is used to initialise the nibble selector IC2A.

IC3d and IC3E buffer the FLM signal generated by the CPU. The frame signal (FR) is automatically derived from FLM through the first flip-flop in IC2 FR toggling at each new frame, which coincides with the falling edge of FLM.

IC3C is responsible for buffering the DISP-OFF control line. IC3C is an inverter (74HC04) which provides automatic disabling of the LCD after a processor reset, since all 8051 port bits become high level on reset.

All data and control signals, as well as the power supply lines, are available for the LCD on connector K1. The pinout given on K1 is valid for the LCD module used in the prototype (salvaged module from 'Supervision' game) and will probably require modification when another LCD model is applied.

The circuit is powered through connector K2. The circuit around the two MAX1044 devices is used to derive Vee (–10 V) from Vcc (+5 V). The MAX1044s are 'charge pumps' that will generate about –10V from Vcc. The charge pump circuit will also help prevent Vee from being present before/after Vcc. This could be a problem with an external negative supply that might take much longer to decay (discharge) than Vcc when power is switched off.

Hardware Implementation

The PCB layout designed for the Low-Cost LCD Controller is shown in **Figure 4**. The LCD controller hardware was connected by short flatcables to an Elektor Flash Micro Board. The required CPU lines (AD0-AD7, RD and Port A signals) were 'tapped' by using a wire-wrap socket for the CPU.

Suitable LCD modules

The prototype used an LCD module salvaged from a Supervision computer game. The module has 160×160 pixels. Onboard drivers could not be identified since they were of the 'bonded die' variety. Other suitable LCD modules include:

– Optrex DMF660N-EW with 240 (w) × 128 (h) pixels. The onboard drivers are Hitachi HD61105 and HD61104 chips. The LCD uses a four-bit parallel data transfer and requires a negative contrast voltage of about –20 V. There are four

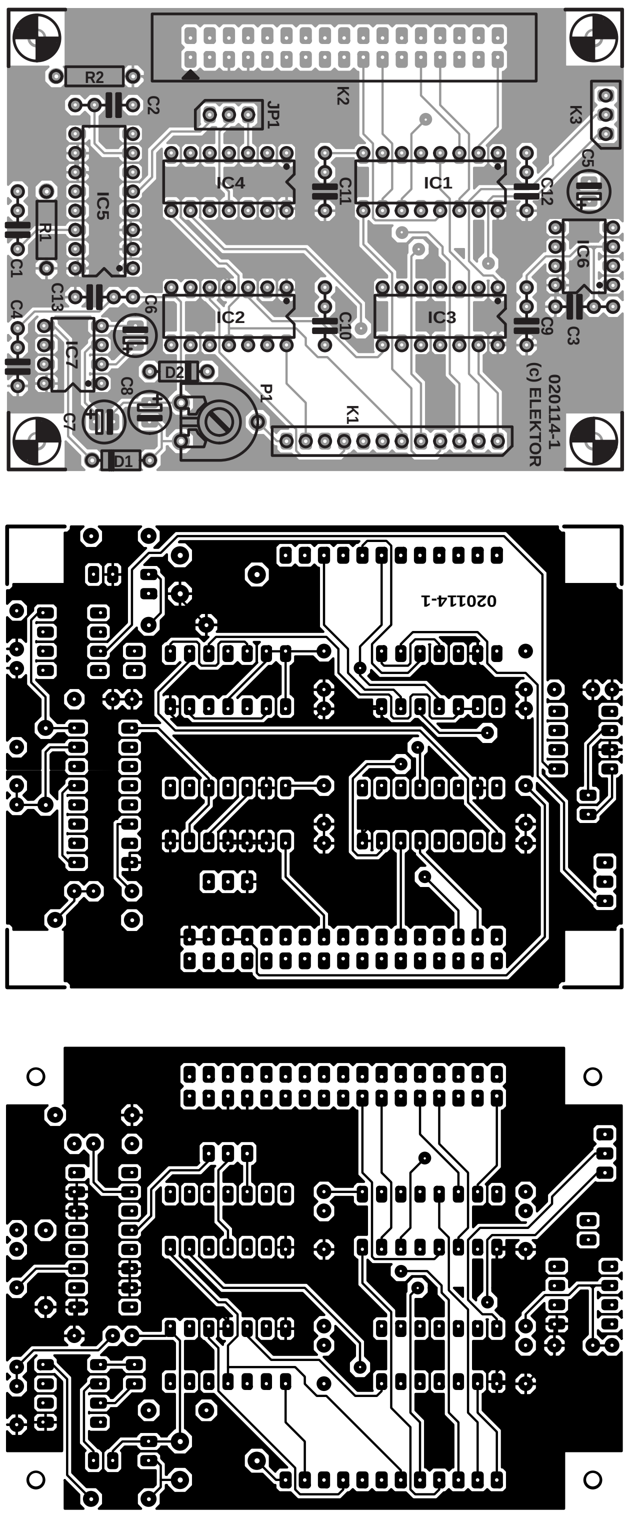


Figure 4. PCB artwork (board available ready-made).

LCD control signals (FLM, CP, LP, and FR, here, called 'M').

– Densitron LM3024 with 240 (w) × 128 (h) pixels.
– Truly MG-160-160-3 with 240 (w) × 128 (h) pixels. The onboard drivers are Samsung KS086.
– Ampire AG240128, also having 240×128 pixels; segment drivers are Toshiba T6A39, common drivers are T6A40.

Obviously, the prototype controller software may need modification(s) when a display with a different resolution is used.

COMPONENTS LIST

Resistors:
R1,R2 = 10kΩ
P1= 10kΩ preset

Capacitors:
C1,C2 = 10pF
C3,C4,C9-C13 = 100nF
C5-C8 = 10µF 16V radial

Semiconductors:
D1 = BAT85
D2 = 12V 400mW zener diode
IC1 = 74HC157
IC2 = 74HC74
IC3 = 74HC04
IC4 = 74HC27
IC5 = 74HC123
IC6,IC7 = MAX1044/CP

Miscellaneous:
JP1 = 3-way header with jumper
K1 = 12-way pinheader
K2 = 34-way boxheader
K3 = 3-way pinheader
PCB, order code **020114-1**, see Readers Services page
Disk, contains all project software, order code **020114-11** or Free Download

Software Implementation

The software for the Low-Cost LCD controller was developed in 8051 assembler. The $\overline{\text{XCSL}}$ signal is derived from the processor's $\overline{\text{RD}}$ signal as explained and the other control signals (FLM, LP and DISP_OFF) are implemented through 'bit banging' of port pins.
The code was developed for the well-known Metalink public domain assembler. This assembler can be downloaded free of charge at www.metaice.com/ASM51/ASM51.htm
It should be relatively easy to port the software to other assemblers —differences are likely to occur with Macro definitions and some other compiler specific directives. The structure of the software is modular. Supporting routines for the serial port of the 8051, utility subroutines and specific control software for the LCD device are all located in separate modules. This allows easier software maintenance and reuse. All modules have the same general structure and consist of two parts: 'Mod_cnst.inc' (constant definitions and variable declarations) and 'Mod_sub.asm' (contains subroutines and program memory constants including tables). The driver software was tested and its use was demonstrated through a number of examples (see section on Main LCD Controller demonstration software).

The software modules will be briefly described below. Details can be found in the source code, which comes with extensive comments and explanations.

Low-level LCD Controller routines

Low level routines are available to initialise the LCD port pins (defined in LCD_cnst.inc), the refresh interrupt routine and the RAM display memory through the subroutine 'LCD_Init'. This routine first sets the correct initial values for the port pins (e.g. $\overline{\text{DISP\_OFF}}$) then initialises memory and refresh datapointers and starts up the refresh Timer interrupt.

Support functions allow the user to scroll or pan the displayed part of the RAM ('LCD_Up', 'LCD_Down', 'LCD_Left' and 'LCD_Right'). Calling subroutine 'LCD_Home' will restore the default setting.

The main task of the LCD controller software is performed by the 'LCD_Hsync_Int' subroutine. This routine is called by the Timer0 interrupt and will be activated every 125 µs, which results in a frame rate of 50 Hz (20 ms) for the 160-row prototype display. During refresh one image row will be updated. The timing diagram in **Figure 5** illustrates how LP is pulled 'low' on the timer interrupt, thus latching the image line loaded during the previous interrupt and enabling the controller hardware to shift in the next image line data. When a complete line has been shifted in, LP is pulled High and the refresh interrupt returns. The refresh routine keeps track of the next row that is to be displayed (variables 'LCD_Ptr_L', 'LCD_Ptr_H') and generates FLM as soon as one complete frame has been displayed (variable 'LCD_Row_Ct'). A separate pointer (variables 'LCD_Crs_L', 'LCD_Crs_H') is used to keep track of the character cursor for the display RAM.

Note that by measuring the duty cycle of LP it is possible to estimate how much performance is used by the Refresh cycle (LP is at Low level) and how much time is still available for the other tasks the CPU should deal with (LP is at High level). For the given set-up LP is logic High for about 25 µs and Low for about 100 µs. This means that LCD refresh costs about 80% of the available CPU performance. An additional advan-

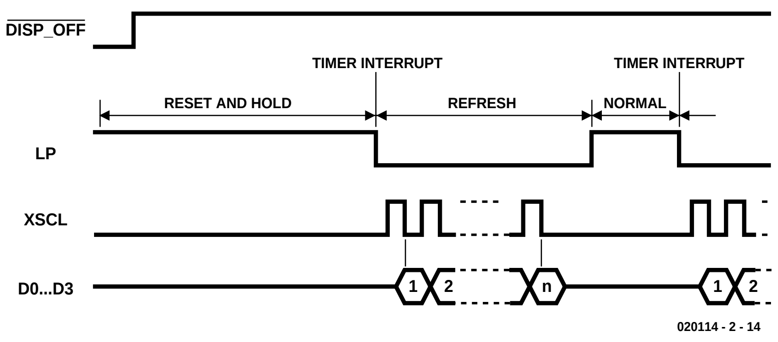


Figure 5. Software Timing diagram.

LCD Module Reverse Engineering

So, you are visiting a surplus store and you've got this great looking matrix LCD in your hands, now what? Usually there are two situations:

– **The display is supplied 'as is' and may be new or salvaged from some piece of equipment.** New displays are often spare parts from repair centres or surplus parts from some system manufacturer. Used parts could still be OK and may have been salvaged from broken equipment (vending machines, Xerox machines or industrial control systems). Unfortunately, you could also run into used defective displays that were replaced by a repair centre. In most cases it will be difficult to repair defective displays. Apart from the obvious broken glass, most likely defects are burned out common or segment drivers. The display will have a number of rows or columns that are either always 'on' or always 'off'. These defects can rarely be detected unless the display is switched on. You have to take a chance here. You can have some confidence in the display if it is still in its original box or wrapped up in sealed anti-static material. Check the display for manufacturer names and type Ids. Also ask for any and all documentation that the store/owner might have.

– **The display is part of some hardware system (either new or used).** Displays that are part of new or used equipment could also be defective! Try to switch on the equipment and check the display for any problems (missing rows or columns). Again, you can have some confidence in the display if the equipment is still in its original box or anti-static material. Check the device and if possible the display for manufacturer names and type IDs. Try to take a peek inside if you can! Also ask for any and all documentation that the store/owner might have.

OK, suppose you are the proud owner of the display, how can you find out about pinouts and other details if you don't have the documentation. Same two situations:

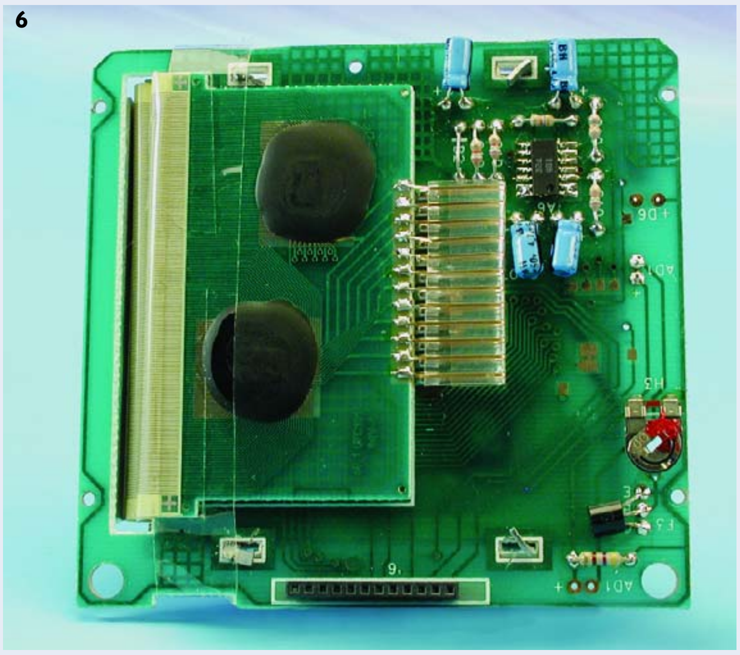
– **The display is supplied 'as is' and may be new or salvaged from some piece of equipment.** Try to identify the make of the display, manufacturer and type number. Then check the Internet for that type. Well-known manufacturers are Seiko-Epson, Optrex, etc. You may not succeed here for several reasons: the display is too old and out of stock or the display was custom-made for the OEM and data is not publicly available. Seiko produces many custom-made displays, which may be recognised by a type number starting with 'ECM' (Epson Custom Made). You may still be able to find some data by using a search engine (e.g., Google or Yahoo) and/or checking newsgroups. Try search engines with the LCD type number, both the complete number and only parts of it.

– **The display is part of some hardware system (either new or used).** Open up the equipment and try not to damage it too much, especially the LCD! Note that you might run into equipment that uses a display connected directly onto the system board. In this case you may not be able to extract the LCD as a standalone module suitable for re-use. Tough luck. On the other hand, you might be fortunate and find an LCD controller (e.g., T6963) or other goodies on the system board that could be re-used also! Once you get access to the display, try to identify it in the same way as described above.

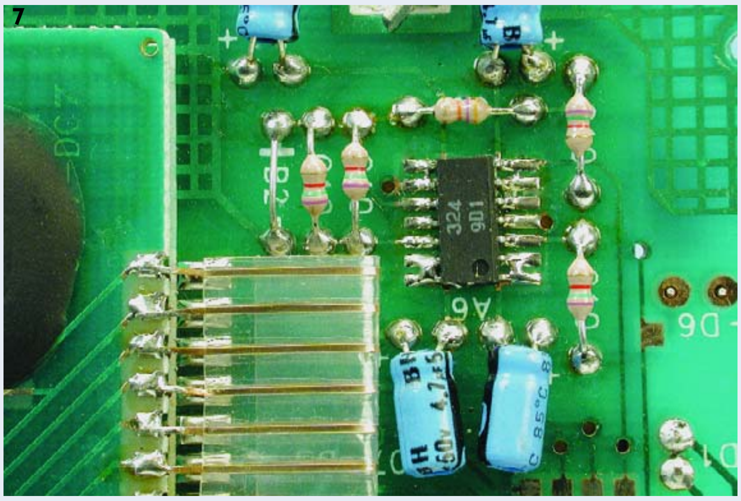
Suppose you did not succeed in identifying the display module or finding a datasheet for it. Again two main courses of action:

– **The display module is supplied 'as is'.** Try to identify the drivers and any other devices on the display module, both manufacturer and type numbers. Then check the Internet for these types. Well-known driver manufacturers are Seiko-Epson, OKI, Hitachi and JRC. You may not succeed here for several reasons: the devices may be too old or the drivers are bonded as bare chips onto the PCB (black blobs of resin on the board). You may still be able to find some data by using a search engine and/or checking newsgroups. Again, try search engines with the type number, both the complete the number and parts of it.
When you have the datasheets for the key components available, it is time to start reverse-engineering the board. Given the general block diagram discussed before and the pin-outs of the drivers, you should be able to identify the external control signals (XSCL, LP etc) and trace them to the connector on the LCD module. The same procedure is used for the bias circuit. Vee voltage levels must be guessed or deduced from driver specs. In general, you can start at a low value and slowly increase Vee until the image is acceptable. Make sure you do not exceed the maximum values given for the drivers. The number of rows and segments can be derived from the driver specs (number of outputs). Note, however, that not all outputs may be used on the last segment or common driver device. Some educated guesswork (trial and error) may be necessary.

6



7



– **The display is part of some hardware system (either new or used).** When you have a functional system, things may be a bit easier. Start by identifying GND and Vcc starting from the pinout of a known device (e.g. SN7400, RAM or EPROM). Switch on the device and use a (dual-channel) oscilloscope to check the signals on the LCD module connector. Given the earlier descriptions of the control signals you should be able to deduce the pinout from the frequencies and/or the timing relations between signals. All digital signals, except for D0-D3, are highly repetitive and should give a stable scope image. Vee and Vadj should also be easy to identify given the voltage levels and/or dependency on the LCD contrast regulator (if any).
When the system is not functional, you can take it apart and follow the approach for a standalone module. However, some reverse engineering on the broken system board might speed up identification of Vcc, Gnd, Vee and Vadj. Things could even be better if the board uses a known LCD controller, which may help to identify all control signals also. Remember that some control signals may not go directly from the controller to the module but pass through some buffer first.

Figure 6 shows the LCD module used for the prototype. The module was part of an old handheld video game, branded 'Supervision'. The game was still operational and was carefully taken apart. The game uses a dedicated CPU that also includes the LCD controller. After identifying GND and Vcc, the game was turned on and a 'scope was used to find the correct pinout for the control signals.
Drivers could not be identified on the LCD since these were bonded chips (notice the blobs of resin). **Figure 7** shows some detail of the bias circuitry, notice the resistors for the voltage divider, the LM324 buffer opamp and the buffer capacitors. The number of rows and columns was deduced to be 160x160 pixels by counting the PCB traces leading to the LCD glass. A magnifying glass and a sharp needle can help you seeing the traces and not loose count!

The method described above is often successful due to the fact that most LCD modules follow the same basic design explained before. Knowing what to expect helps solving the puzzle. You have some room for trial-and-error in connecting the control- and data-signals, but usually, the display will not tolerate Vee/Vadj on any of its control pins. So, pay attention to correctly identifying Vee/Vadj and then double check before hooking up the display!

tage of the Dallas 89C420, apart from its much higher speed, is that the device has improved DPTR instructions (i.e., auto-increment and secondary DPTR) that would again reduce the relative performance cost of the display refresh cycles. Note that the prototype software needs to be modified to take advantage of these features.

High-level LCD Controller routines

Higher-level LCD routines are available that write one character on the display at the current cursor location ('LCD_PutChar') and routines that write a 'zero terminated' constant or variable string ('LCD_PutStrCnst' and 'LCD_PutStrBuf'). The cursor will automatically increment, but does not wrap around to the next line. The cursor may be set by the 'LCD_GoTo' routine. Characters are defined as an 8x8 pixel bitmap in the 'Chr_sub.asm' file.

The subroutine 'LCD_Load' will take a pointer to a ROM address and load a predefined image into the display RAM. One example is given in 'LCD_Girl.asm'. During an image load, when data is moved from code ROM to display RAM, the LCD refresh is turned off to speed up the operation. The display refresh may be turned on or off at any time by calling 'LCD_On' or 'LCD_Off'.

Serial Port Interface

The serial port software (files 'ser_cnst.inc' and 'ser_sub.asm') provides initialisation of the 8051/80535 serial port hardware. Default baudrate is 9600. Assembler directives are available to select 4800 or 9600 baud and select the CPU clock frequency (11.0592 MHz or 12.000 MHz). Basic routines allow character transmission and reception and string transmission to a terminal program (e.g. hyperlink) connected to the serial port.

Utility (UTL) software

The UTL module ('utl_cnst.inc' and 'utl_sub.asm') provides general support functions like ASCII to Hex conversion, delay routines, etc.

Character software

This module ('chr_cnst.inc' and 'chr_sub.asm') provides predefined bitmaps for characters/symbols that may be loaded into the LCD memory to display characters and text strings. The character symbols are stored as 8×8 bitmaps and resemble the old VGA font. Note that the example software uses additional bit-swapping routines (i.e. databit *n* <=> databit (8-*n*)) to allow adaptation of available bitmap patterns to the LCD hardware.

Bitmap software

This module ('lcd_girl.asm') provides a predefined bitmap for an image that may be loaded into the LCD memory. The 160×160 pixel image is stored as hex bytes and was derived from a Windows .BMP file.

It is relatively straightforward to add routines that support typical graphics functions like line drawing. Drawing routines for commonly seen LCD controllers like the SED1330 can be used after minor adaptations. The major difference is that our displayed image is a section of microcontroller memory instead of dedicated LCD controller memory. Examples of 80C51 assembler code for graphics routines (e.g., Bresenham line-drawing algorithm) may be found on the Internet.

Main LCD Controller demonstration routines

The main software module called 'Tst51LCD.asm' contains the actual application employing the #include assembler directive to incorporate the necessary support modules. Default settings for the 'Tst51LCD' software are suitable for the CompactFlash-Board. If you want to assemble the source for the High-Speed Controller Board, change line 43 into

```
FLASH   EQU UNDEFINED
```

and change line 42 into

```
DALLAS  EQU DEFINED
```

Note that the software was tested with the High-Speed Controller Board running at 27 MHz. If you want to use the software at a different clock speed, you will have to change the intialisation code for the serial port. Also, you will need to insert some NOP operations in the macro LCD_LOAD_X, to make sure the LCD's timing is met.
Selection of a specific processor type (e.g., 8051 or 80C535) and other modes/options (e.g., baudrate) are possible by activating assembler directives in 'Tst51LCD' or one of its #included modules.
The 'Tst51LCD' software first initialises the CPU (serial port, stack-pointer and so on), then initialises the LCD, starts up the refresh operations and allows the user to enter simple commands that will demonstrate the main features like pan, scroll, image- and string display.
The software for this project may be downloaded from the *Elektor Electronics* website. It contains the necessary source code, as well as two HEX files. FLASH.HEX is intended for direct use with the Flash Micro Board, while DALLAS.HEX is for the High-Speed Board.

Conclusion and Final Remarks

The Low-cost LCD Controller described in these two article instalments provides a second life to surplus display gems and adds new features to a standard 8051 design with minimal additional hardware. Have fun and let use know about your experiments with matrix LCDs and/or your ideas for improvements!

(020114-2)

References

AT89S8252 Flash Micro Board, *Elektor Electronics* December 2001.

High-Speed Controller Board, *Elektor Electronics* May and September 2002.

www.seiko-instruments.de/

www.optrex.co.jp/us/product/catalog/index.html

www.eio.com/public/lc

Free Downloads

8051 source code, Hex files for Elektor Flash Micro and High-Speed Controller Board. File number: **020114-11.zip**
PCB layout in PDF format. File number: **020114-1.zip**
www.elektor-electronics.co.uk/dl/dl.htm,
select month of publication

Need DC Power? Try a PC/AT supply!

Modifications for non-standard output voltages

By J. Waegebaert

Although the amount of power drawn by electronic circuits is steadily decreasing, there are still many devices that need a large amount of low-voltage power. Some examples are miniature (PCB) drills, model railways, vintage radios, safe low-voltage lighting and heating systems, battery chargers and so on. With a few modifications, a standard PC/AT supply is an excellent choice for the job.

When they need a low-voltage power supply, most hobbyists use the 'classic' transformer/bridge rectifier combination. This has many advantages, such as simple design, outstanding mains isolation, robustness and low interference levels. However, there are also a few serious disadvantages to this approach: the supply is large and heavy, it is difficult to efficiently obtain an adjustable voltage at high power levels, and it is fairly expensive.

Naturally, the alternative is a primary-switching converter (off-line switcher). For a hobbyist, designing such a power supply from scratch or copying an existing design is expensive, impractical and unsafe. It is expensive because in most cases, any error is punished by having the switching transistor go up in smoke; it is impractical because many of the necessary components are simply not available from retail sources; and it is unsafe because the primary side is directly connected to the mains. To this can be added an (unjustified) fear of everything composed of a core and a few turns of copper wire.

In this article, we describe an inexpensive solution to the above problem, based on using a standard PC/AT power supply. These are mass-production items that are available everywhere, and if necessary, one can be salvaged from a discarded PC or picked up for next to nothing at rallies and jumble sales. Everything you need to build a primary-switched supply is already there.

PC/AT supplies

The design of a PC/AT supply is quite conventional. It consists of a 'half-bridge' converter. The mains input voltage is directly rectified to generate a DC voltage of 320 V (**Figure 1**). Half of this DC voltage is applied to the primary winding of transformer T1 via a switching transistor. By alternately switching Q1 and Q2, the transformer is magnetised in both directions. This means that the energy transfer is very similar to that in a traditional transformer/rectifier design. On the secondary side, the windings have a common centre tap, and the voltages are full-wave rectified by two diodes for each output and then filtered. This is done by D1 and D2 for 12 V, and by D3 and D4 for 5 V. The four remaining diodes are used to generate the negative voltages. In most AT supplies, these negative voltages are completely unregulated, and they can be literally regarded as 'extra' voltages.

The base currents for the switching transistors flow via control transformer T2, so the secondary side is fully electrically isolated from the primary side. The primary side of the power supply is limited to the circuitry around transistors Q1 and Q2, capacitors C1 and C2, transformers T1 and T2 and so on. **Be careful – this portion of the supply is connected to the mains.** Also, bear in mind that Figure 1 is a basic schematic diagram, in which a number of networks have been omitted for the sake of clarity.

Even today, all of the drive signals for the switching transistors in a PC supply are provided by an IC dating from the early days of switching power supplies, the Texas Instruments TL494. Due to its age, this IC is not exactly the best choice for building a supply with the highest possible performance, but it results in a simple design that is ideal for our purposes, since it is easy to modify.

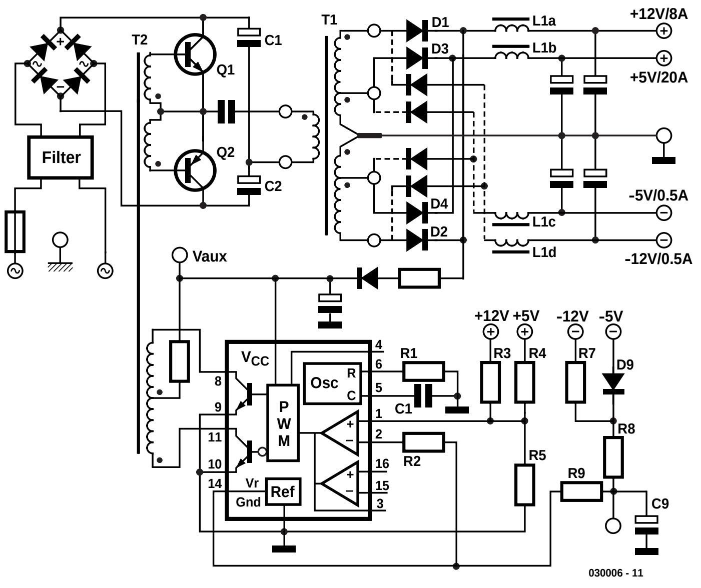


Figure 1. Basic schematic diagram of a PC/AT power supply.

The switching frequency is approximately 30 kHz, which is determined by the values of R1 and C1. The conduction time of Q1 and Q2 is continuously adjusted during each 33-μs period. This time ultimately determines the value of the output voltage, since the duty cycle of the square wave on the cathodes of the diodes (D1 and D2) depends on the conduction time (see trace 2 in **Figure 2**). The DC voltage at the output is (approximately) equal to the average value of the square wave. Via a voltage divider (R3–R5), the TL494 compares the voltages on the 5-V and 12-V outputs with its internal 5-V reference voltage. If the output voltage is too low, the conduction time of the switching transistors is increased, thus increasing the output voltage.

Modifications

A typical 200-W AT supply has the following output specifications:

+5 V,	3–22 A
+12 V,	0.5–8 A
–12 V,	0.5 A
–5 V,	0.5 A

This gives an idea of the loads that the components in the various secondary circuits can handle (transformer, diodes and filter choke). When modifying a supply to suit your purposes, you must keep within a number of limits, as follows:

- The total secondary power must never exceed 200 W. Transformer T1 and transistors Q1 and Q2 are specified for this value.
- The current through each of the 12-V windings must never be greater than 4 A. This is determined by the specifications of L1a, D1 and D2.
- The current through each of the 5-V windings must never be greater than 10 A. This is determined by L1b, D3 and D4.
- The transformer must be symmetrically loaded for the two magnetic polarisations. This means that full-wave rectification must always be used.
- The desired output voltage must be in the neighbourhood of ±5 V or ±12 V. This is because transformer T1 has a certain turns ratio that must be respected. Deviations of up to ±30% (3.5–6.5 V and 9–15 V) are certainly possible. However, the ratio of the output voltages always remains the same. If 6 V is generated on the 5-V output, the voltage on the 12-V output will be 14.4 V (6 × 12.5 ÷ 5).

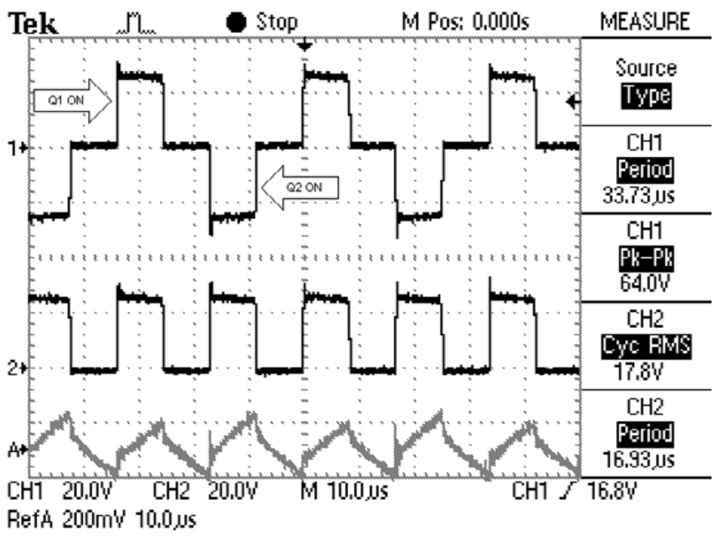


Figure 2. Typical secondary waveforms.

Example 1: 6 V / 16 A

The desired value of 6 V is sufficiently close to 5 V. The desired current requires 8 A from each arm of the rectifier, which is less than the available amount of 10 A, and the total power is 96 W. This objective is thus feasible. The +12-V, –12-V and –5-V outputs are not necessary and can be deleted. This yields the circuit shown in **Figure 3**. D1 and D2 are only necessary to produce V_{aux}, which powers the TL494.

The output voltage is determined by R4 and R5. The TL494 regulates the output voltage such that the voltages on pins 1 and 2 are the same. The voltage on pin 2 is equal to the reference voltage (5 V), so the voltage on R5 is also 5 V. This means that there is 1 V on R4. The value of R4 is then:

R4 = (6 V – 5 V) ÷ (5 V) × R5

If we choose a value of 4k7 for R5 and 1k for R4, the output voltage will be 6.06 V.

Example 2: 24 V / 4 A

An output voltage of 24 V does not fall within the mandatory ±30% range. This means that we have to take a different approach in order to achieve this 'high' output voltage. A possible solution is shown in **Figure 4**.

Here diodes D5 and D6 have been added to the rectifier. These diodes must also be power diodes of the same type as D1 and D2 (which means they cannot be the diodes used to produce the –12 V / 0.5 A output in the original supply). The 24-V load is actually connected between the +12-V and –12-V outputs, which have been renamed to +24 V and 0 V. The output voltage is regulated using the +12-V output according to the formula:

R3 = (12 V – 5 V) ÷ (5 V) × R5

If R5 is 3k3, R3 becomes 4k7 and the output voltage is 24.2 V (2 × 12.1 V).

Although the –12-V circuit is not explicitly included in the control loop, in practice it will almost exactly track the positive output, due to the excellent coupling between the secondary windings of T1 and the coupled choke L1. The waveforms shown in Figure 2 were measured using this configuration with a 3-A load.

Other configurations

In some PC supplies, both secondary windings of T1 have the same wire diameter. This means that the maximum current load on the 12-V winding can also be as much as 10 A, as long as the total power of 200 W is not exceeded. With such a supply, the load on the 24-V output in Example 2 above could be as much as 8.3 A (200 W ÷ 24 V), assuming that the diodes and choke are modified accordingly.

If the 5-V winding is also used with the circuit configuration shown in Figure 4, 10 V / 10 A is also available, and so on. Naturally, the arrangement shown in Figure 4 can also be used if it is desired to have ±12 V / 2 × 4 A or ±5 V / 2 × 10 A. There are thus many different possibilities.

Component modifications

Filter choke L1

In an AT supply, choke L1 usually has five bifilar windings. Due to the large current flowing through L1b, two windings are used for this branch.

The five sections of L1 form a coupled inductor. This means that the turns ratio for the 5-V and 12-V windings is the same as the voltage ratio. The opposing directions of current flow for the positive and negative voltages must also result in balanced polarisation of the magnetic field. The choke sections on the negative side of the supply are thus connected in the opposite direction.

In Example 1, L1 can be used as is without any problems. In Example 2, this is not possible, since the original –12-V winding on the choke (L1d) is usually wound using thinner wire than L1a. If this is the case, L1d is probably only designed to handle the specified 0.5 A, which makes it unsuitable for the 4-A load of Example 2.

The solution to this problem is to rewind choke L1. First, remove all the windings while carefully noting the exact number of turns for each winding. After this, you can rewind the choke with only two windings, each of which has the same number of turns as L1a and a wire diameter at least as great as that of L1a, so that can be used with a 4-A load. The two windings must be wound bifilar (together) on the core. Spread the windings evenly over the entire core. When reconnecting the choke, make sure the polarities are correct.

If you have to wind a new choke for a different output voltage, it’s a good idea to adjust the number of turns. For example, for 15 V the number of turns should be increased by a factor of 1.25 (15 ÷ 12). Note that this is not strictly necessary. If the number of turns is too small, the ripple voltage on the output will be

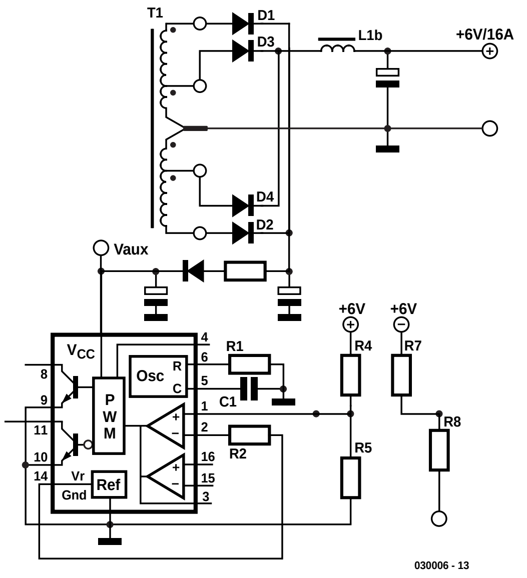


Figure 3. A configuration for 6 V / 16 A.

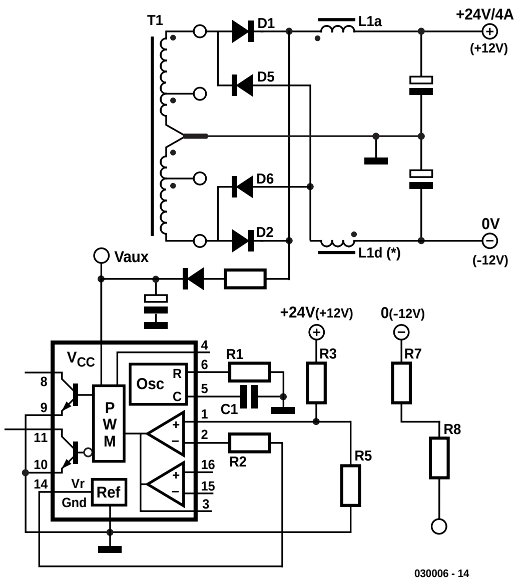


Figure 4. This arrangement allows the supply to provide 24 V / 4 A.

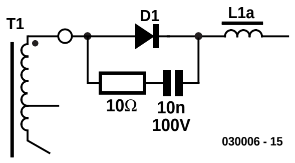


Figure 5. Diode snubber network for suppressing voltage spikes.

somewhat greater.

In Example 2, it is also possible to use the two sections of L1b (since each winding is suitable for 10A). Although this yields are relatively large ripple voltage (see trace RefA in Figure 2), it is ideal for a quick test.

Diodes

Fast Schottky diodes must be used for the rectifiers. These diodes are usually arranged in pairs in a TO220 package, with both cathodes connected to the middle pin. They can handle 'enormous' currents (2 × 25A is not exceptional), but they are somewhat less tolerant with regard to maximum reverse voltage (which is sometimes as low as 25 V). The measured signals in Figure 2 show that the peak-to-peak voltage on the 12-V winding is 64 V! This high voltage is applied to the reverse-biased diodes. To obtain a reasonable margin, it is necessary to ensure that these diodes have a reverse voltage rating of at least 90 V, even if the output voltage is only something like 10 V. This is because the voltage is controlled by adjusting the width of the pulse, rather than its amplitude. Reducing the voltage from 12 V to 10 V has almost no effect on the amplitude of the square-wave signal. For the 5-V winding, diodes with a reverse voltage rating of 40 V must be used.

In addition, it is advisable to connect an RC snubber network in parallel with each diode (see **Figure 5**). This reduces the values of the voltage spikes generated when the diode starts to conduct or stops conducting. Be sure to use a capacitor with an adequate working voltage!

The diodes in the 5-V circuit can be reused for almost all feasible configurations, since they are rated for 45 V / 2 × 10 A. The 6-V / 16 A supply of Example 1 thus presents no problems in this regard.

Things are different with the 12-V circuit. In some supplies, this circuit is fitted with PR3002 diodes, which are specified for 100 V / 3 A. Evidently, manufacturers of (inexpensive) power supplies assume that the maximum continuous current will never be greater than 6 A. If your configuration demands more, a better choice of diode is the Philips PBYR20100CP (100 V / 2 × 10 A), for example, or possibly the PBYR10100 (100 V / 10 A). Incidentally, these types can also be cooled better, thanks to their TO220 packages.

Capacitors

The electrolytic capacitors at the outputs have to handle the ripple currents. Due to the internal resistance of the capacitors (ESR), these currents produce a certain amount of ripple voltage at the outputs (trace RefA in Figure 2). The lower the ESR, the lower the ripple voltage and the lower the temperature of the capacitor. This means that only low-ESR types can be used here. A second consideration is the working voltage. The capacitors in the supply usually have values of 2200 μF / 10 V for the 5-V output and 1000 μF / 16 V for the 12-V output. If output voltages greater than 8 V or 14 V are desired, the capacitors must be replaced by low-ESR types with working voltages of 16 V and 25 V.

Transformer T1

The ground leads of the transformer windings, which form the centre tap, emerge from the transformer as a 'pigtail' and are thus not connected to a terminal. The ends of the windings are twisted together, and they can be easily separated to yield two independent windings with asymmetric 5-V taps. This makes it possible to build supplies having two electrically isolated outputs. However, note that the secondaries of T1 are wound bifilar, so the insulation voltage is limited and is only suitable for isolating low-voltage portions of circuits. It is not suitable for isolating the mains voltage, for example.

Ensure that the transformer is always symmetrically loaded for the two halves of the magnetisation waveform. This means that full-wave bridge rectification must always be used, since there is no longer a centre tap available (**Figure 6**). Only the voltage on the second output is regulated.

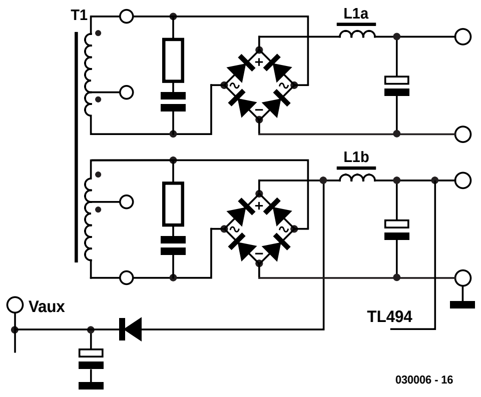


Figure 6. Two electrically isolated outputs.

A nominal voltage of 7 V is available on the winding between the 5-V and 12-V terminals (with the two loose ends of the pigtail not being used). This can be used to generate output voltages of around 7 V ± 30% (5–9 V or 10–18 V).

Other details

TL494 supply (Vaux)

Naturally, the switch-mode controller also needs a source of power. This is derived from the peak secondary voltage, which means that around 20 V is available. After modifying the supply, measure this voltage. The allowable working range for the TL494 is approximately 7–40 V.

Power Good detection

A PC supply has a 'Power Good' output. This output is high (+5 V) if all voltages are correct. If the supply is used in one of the configurations described above, the detection line will naturally indicate a fault. In this case, most supplies shut down the TL494 via pin 4. This means that this circuit must also be modified.

The detection circuit of just about all PC/AT supplies is based on a small network similar to the network formed by R7–R9, D9 and C9 in Figure 1. If everything is OK, the voltage on C9 is usually around 3 V. If you want to reproduce this situation without actually implementing a true Power Good function, you can simply remove R8 and connect an extra resistor across C9, with a value such that 3 V will be present at the junction. C9 must be left in the circuit, since it provides the soft-start function for the entire power supply.

This means that you will have to track down this network in the supply before removing any other items. It is easy to find, since it forms the only connection between the –5-V and –12-V outputs and the control circuit. Once you have located the network, use a multimeter to measure the voltage at the junction. **When making this measurement, avoid touching any components carrying mains voltage!**

Overload detection

Two different types of overload detection are commonly used in AT supplies. Older designs use a small current transformer in series with the primary winding of T1. After rectification and filtering, the signal bearing the information about the magnitude of the current is monitored by the second comparator of the TL494 (pins 15 & 16). In the event of an overload, the pulse-width drive to the switching transistors is reduced. This system is foolproof and provides secure overload protection.

Newer designs dispense with exact measurement and deduce the current from the duty cycle. If the duty cycle is too large, it is reduced by the second comparator. This makes it unnecessary to use a current transformer (thus reducing the cost of the supply). In most cases, the protection is adequate. However, if the transformer becomes (briefly) saturated for some reason, it cannot be detected using this method, although it can be detected using a current transformer.

Minimum load

If you examine Figure 2, it is clear that the output capacitors will charge to the peak value of the square-wave voltage on the cathodes of the diodes if no load is connected. Consequently, a certain minimum load is always necessary. You might want to use a nostalgic incandescent lamp for the power-on indicator instead of an LED, or you can connect a fan to the output. Of course, you can always simply connect a load resistor across the output terminals.

Temperature

The rectifier diodes become quite warm with currents of this magnitude. At 10 A, the voltage across a Schottky diode is 0.4–0.6 V, which yields a power dissipation of around 5 W (this is an approximation, since the diode does not conduct all the time, but there are still two diodes in a single package). A TO220 package has a thermal resistance of 50 °C/W (junction to ambient). If no heat sink is used, 5 W will thus cause a temperature rise of 250 °C. This means that a heat sink is imperative.

With the same TO220 package, the thermal resistance from the junction to the case is 1 °C/W. If the heat sink has a thermal resistance of 10 °C/W, for example, the total thermal resistance is 11 °C/W. At 5 W dissipation, this causes the temperature to rise by only 55 °C. If the ambient temperature is 30 °C, the junction temperature will be 85 °C, which is acceptable.

In accordance with the motto ‘if it’s warm, you have to blow’, it is naturally possible to use a fan. With a reasonable air flow (> 0.5 m/s), the original thermal resistance of the heat sink is reduced by factor of approximately 3. In our example, this gives a total of 4 °C/W. This yields a temperature rise of 20 °C instead of 55 °C, which is naturally better.

The switching transistors are also fitted to a heat sink, which is already adequate. **Lethal voltages are present on this heat sink,** so don’t just give it a quick finger test to see whether the temperature has become too high!

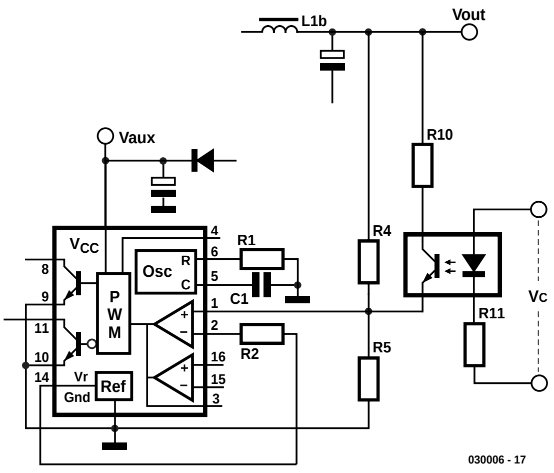


Figure 7. Output voltage adjustment can be obtained by adding R10 and an optocoupler.

Other uses

This type of supply is not especially suitable for use as an adjustable supply, but it is still possible to build in a certain amount of adjustment range by modifying the feedback loop for the TL494. In this way, the supply can be used to control the speed of a drill or the brightness of a string of lamps. As can be seen in **Figure 7**, R10 can be connected to modify the resistance of R4 in an adjustable manner. This affects the division ratio for the output voltage. The parallel resistance can controlled by a normal transistor or an optocoupler in series with R10. If the current through the LED of the optocoupler is increased, the output voltage will decrease.

Have a go!

As we have shown here, a PC/AT power supply can easily be modified to generate or control relatively large amounts of DC power.

Before you start making any changes, it’s a good idea to first trace the circuitry around the TL494 in order to locate the resistors for the feedback and Power Good networks. After this, in many cases you can use the suggestions presented here to build a power supply that provides the desired voltage and current.

Finally, we would like to warn all of you keen experimenters to be extremely careful when opening the enclosure of a PC power supply. **Mains voltage is present on a number of components, and that can have fatal consequences if you touch these components!**

(030006-1)

References
TI TL494 datasheet:
www-s.ti.com/sc/ds/tl494.pdf
Philips PBYR20100 datasheet:
www.semiconductors.philips.com/pip/PBYR20100CT.html
Philips Thermal Management APN:
www.semiconductors.philips.com/acrobat/applicationnotes/APPCHP7.pdf

C Compilers for the PIC Micro

Efficiency from a higher programming language

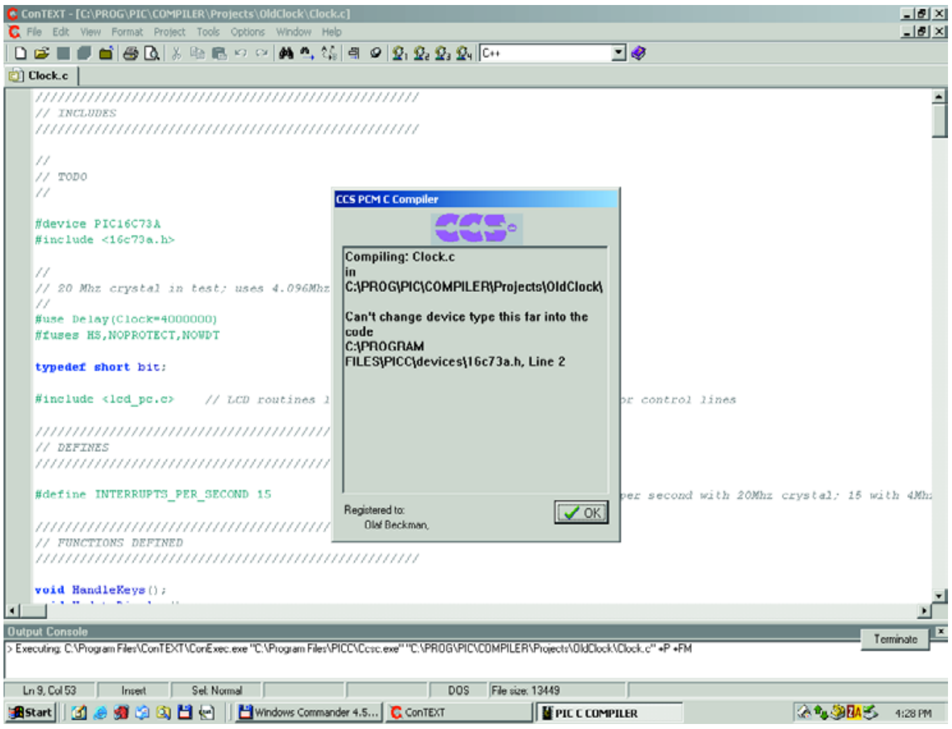
By O. Beckman Lapré

The PIC microcontroller in its many guises is found at the heart of countless electronic projects and designs (including those from Elektor). Most software developers use brick-by-brick assembly code to write programs for the PIC. However, it is also possible to program these wonderful micros using a higher programming language like 'C' which rightly became an industry standard.

The tremendous popularity of the C programming language is largely due to it having been designed, right from the start, for better access to the system hardware. Also, C is not a bad performer when it comes to code efficiency. These days, C allows you to develop almost anything for microcontrollers (from the smallest PIC to X-scale 'monsters'), digital signal processors, operating system programs for Windows or Linux, right up to UNIX and mainframes, not forgetting administrative programs.

C is also great for building device drivers and even complete operating systems (including Linux and Windows 2000/XP), and of course application programs we use every day (from Word to Mozilla). Very likely, C is currently the most universally applicable higher programming language — and likely to remain for some time to come. It pays to learn C!

In this article we look at three C compilers for the Microchip PIC series of microcontrollers. In electronics hobby land, the PIC currently ranks among the most popular microcontrollers, although the competition from the Atmel AVR side is fierce. The PIC is easy to apply in



```
////////////////////////////////////////////////////
// INCLUDES
////////////////////////////////////////////////////

//
// TODO
//

#device PIC16C73A
#include <16c73a.h>

//
// 20 Mhz crystal in test; uses 4.096Mhz
//
#use Delay(Clock=4000000)
#fuses HS,NOPROTECT,NOWDT

typedef short bit;

#include <lcd_pc.c>    // LCD routines l                          or control lines

//////////////////////////////////////
// DEFINES
//////////////////////////////////////

#define INTERRUPTS_PER_SECOND 15                       per second with 20Mhz crystal; 15 with 4Mh

//////////////////////////////////////
// FUNCTIONS DEFINED
////////////////////////////////////////////////////

void HandleKeys();
```

```
Compiling: Clock.c
in
C:\PROG\PIC\COMPILER\Projects\OldClock\

Can't change device type this far into the
code
C:\PROGRAM
FILES\PICC\devices\16c73a.h, Line 2
```

```
> Executing: C:\Program Files\ConTEXT\ConExec.exe "C:\Program Files\PICC\Ccsc.exe" "C:\PROG\PIC\COMPILER\Projects\OldClock\Clock.c" +P +FM
```

Figure 1. The standard CS compiler in combination with the ConTEXT program editor.

practice and often just as good as the industry-standard (and up-market) 8051 controller.

Hobbyists typically have their own requirements for a C compiler, like ease of use and low cost (the latter to avoid lengthy explanations to the missus). Three compilers that certainly meet these requirements are ‘PCM compiler’ from CCS, ‘C2C compiler’ by Pavel Baranov, and the ‘C for PIC Microcontrollers’ from Matrix Multimedia. Although these products differ considerably in practical use and features, they certainly meet the requirements of the home programmer. The product versions discussed here are probably not the latest and may differ in some respects from the most up to date releases.

Development suite

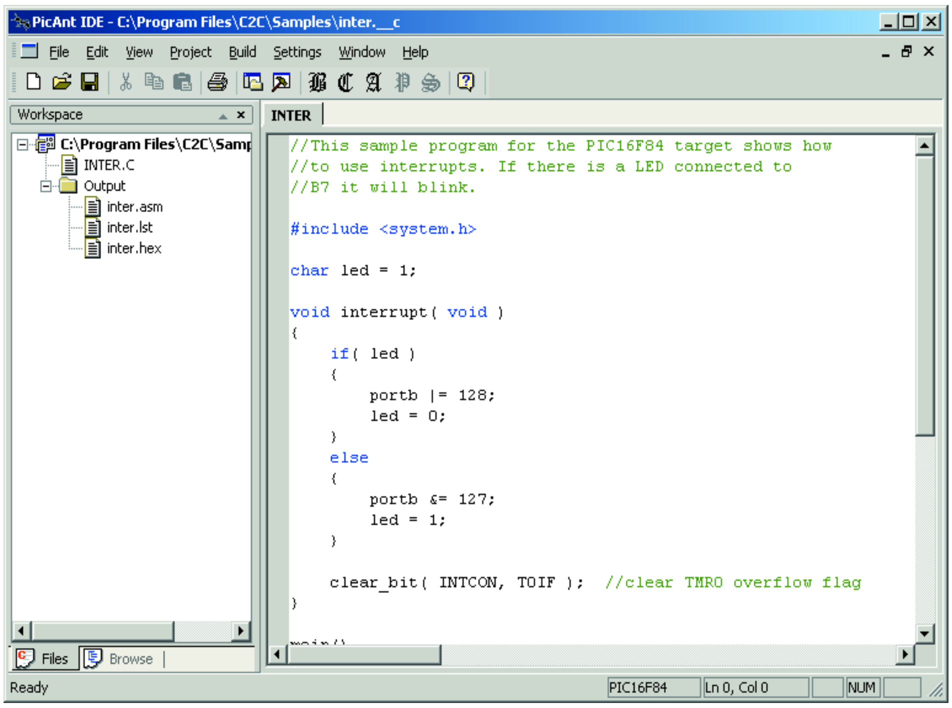
If you do a lot of microcontroller code building, ease of use is a primary requirement. Also, the beginning programmer should not have to wade through hundreds of pages in the User Manual to learn the basics. All three C compilers assume a good command of C. Especially in the case of PCM and C2C, the relevant documentation on the language itself is rather scanty and mainly intended to describe the possibilities, impossibilities and the library routines.

The compilers show significant differences when comparing their development surroundings.

The standard CSS compiler is, well, a compiler and actually works ‘alone’. It has to be invoked through a DOS window or a console command line that includes the name of the file options, if any. Obviously, it would be far more user friendly if the compiler could be launched from an editor. For that purpose, the program (text) editor ConTEXT is just fine (see **Figure 1** and the web reference at the end of this article).

The C2C compiler (version 4.0e, see **Figure 2**) has a Windows-based platform around it and is slightly easier to use than PCM. A disadvantage, however, was found in a number of bugs in the program that caused text to be displayed erroneously (on occasions, and in the version available to the author). Although this caused problems at times, the program did dot crash.

The C2C compiler optionally allows you to program the PIC simply by calling a programming utility.



```
//This sample program for the PIC16F84 target shows how
//to use interrupts. If there is a LED connected to
//B7 it will blink.

#include <system.h>

char led = 1;

void interrupt( void )
{
    if( led )
    {
        portb |= 128;
        led = 0;
    }
    else
    {
        portb &= 127;
        led = 1;
    }

    clear_bit( INTCON, TOIF );  //clear TMRO overflow flag
}
```

Figure 2. De C2C compiler has a Windows-based development suite for C programs.

Listing 1.

Structure ‘mapped’ to an I/O port

Example for PCM compiler:

```
struct _keys
{
bit mode_sw;   // LSB
bit plus_sw;
bit minus_sw;
bit set_sw;
bit unused : 4;
} keys

#byte keys = 5   // ‘keys’ struct is mapped to address 5 = PORTB

if (keys.mode_sw == 1)
{
// mode switch was pressed
}
```

With the C2C compiler, the port first has to be read. Next, the ‘&’ operator is used to test if a bit is set or not, as in the example below:

```
char keys@5      // keys are mapped to address 5 = PORTB

if (keys & 1)
{
// mode switch was pressed
}
```

Internet addresses:

C for PIC Microcontrollers. CD-ROM product, price from £39:
www.matrixmultimedia.co.uk

PCB compiler (for 12-bits PICs), PCM compiler (for 14-bits PICs), PCH compiler (for 18-bits PICs) and PCW(H) compiler (Windows version):
www.ccsinfo.com/picc.shtml
Price: from $125 + P&P

PCW demo version (1 k):
www.ccsinfo.com/demo.htm

C2C compiler:
www.picant.com/c2c/c.html
Registration: $ 59

ConTEXT programming editor:
www.fixedsys.com/context/

C2C and PCM are capable of producing a .LST file that enables the generated assembly code to be checked by the compiler. That is often used as a last resort when you've discovered that the code does not do what you want it to, or to tweak the efficiency of the code. In some cases, it may be worthwhile to apply a different algorithm to see if it improved upon the compiler efficiency. Fortunately, this is not normally required. However, the author came across cases where problems could be solved by disabling the compiler's code optimiser feature.

The CSS compiler comes in no fewer than five different versions. The PCB version is intended for 12-bit PICs. The PCM variant we've mentioned so far is for 14-bit PICs, while the most recent version, PCH, handles 18-bit PIC micros. All program versions are Windows command line applications, which, on being invoked, pop up a dialogue showing the progress of the compilation process. As already mentioned, their functionality needs to be completed with a programming editor like ConTEXT. We also came across real Windows versions (called PCW and PCHW), but they are far more expensive than their command line counterparts.

The C2C compiler is distributed as shareware and a lot cheaper than the CSS products. New releases keep appearing. The author tested versions 4.0 and 4.6 but in the mean time a major new version, 5.0, appeared, consisting of a complete IDE called PicAntIDE.

The excellent C for PIC Micros is supplied on CD-ROM only, and is competitively priced at £39 for the student version. The current version is 2.0.

Library

When comparing the programs, major differences are also observed in the 'library' department as well as in the possibilities of the C language. For example, the C2C compiler has a 'tighter' link with the hardware because that is handled at a low level. By contrast, with the PCM compiler the emphasis is clearly on a quick start involving a minimum amount of programming effort.

Many of the library routines incorporated into the PCM compiler are also available on the C2C complier website, which was written by a third party. In both cases, a built-in library is available for RS232 and delays.

The advantage of the PCM compiler is its ability to support structures and bit fields. For example, a structure may be 'mapped' directly onto an I/O port with the ability to control individual port lines, as illustrated in the code example in **Listing 1**.

The PCM compiler comes with an extensive library for the I^{2}C and RS232C serial communications protocols and there's even HD44780 LCD controller software, including an extensive print(f) command. The output of the print(f) routine, by the way, is subject to adaptation using a user-defined putc() routine that's employed by print(f). In this way, print(f) output may be directed to a graphic LCD just by writing a putc() routine for your particular LCD (in other words, a kind of device driver). A large number of exotic drivers is supplied for countless applications like touchscreen control and temperature sensor reading. These, however, require ICs to be used in the application circuits that are actually controlled via the available drivers. In all other cases, you have to 'write your own'!

As opposed to the C2C compiler, PCM also supports 32-bit IEEE floating point routines. Fortunately, that need not be a problem in all cases because the PIC is usually applied in relatively simple control systems. All compilers cheerfully handle 8-bit and 16-bit variables.

Further outlook

All three compilers represent excellent value for money and are perfect for home programmers, although it must be said that C2C is a bit more aimed at the advanced programmer. The PCM compiler has more to offer than the C2C product (floating point, bit fields) and comes with an extensive library (for a really quick start) but on the down side costs twice as much as C2C. C for PIC Micros best covers the educational side, i.e., learning to program in C, and is remarkable for its broad range, comprehensiveness and relatively low price.

Finally, programmers as well as hardware designers should ensure that the target circuit is suitable for in-system programming (ISP), allowing the cycle consisting of compiling, chip erasure, programming and debugging to be kept as short as possible.

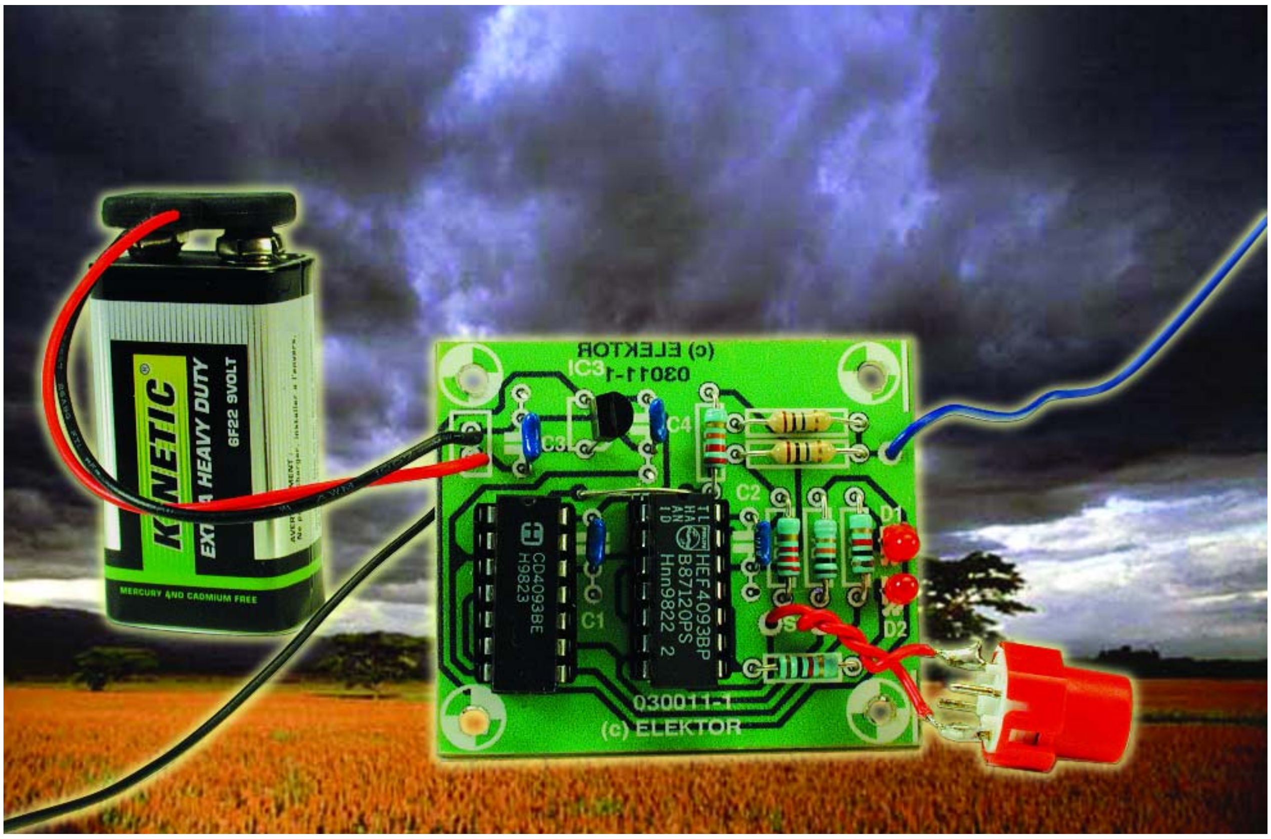
(030098-1)

Lightning Intensity Detector

An indicator for extremely high voltages

From an idea by B. Oehlerking

It is generally known that the forces of nature can be destructive, as proved time and again by storms, tsunamis, tornados, tidal waves and earthquakes to mention but a few. Fortunately, the occurrence of these awe-inspiring natural forces is rare. The thunderstorm, a more frequently occurring phenomenon with less impact than any of the above (but by no means less dangerous), is caused by electricity (at an enormous scale), which makes it suitable for detection by electronic means.



The author got the idea to design this circuit during an outdoor sports event. Many among the audience were quick to deploy their umbrellas when it started to rain. This happened while a distant sound of thunder could be heard. All of a sudden, a large group of people appeared to throw their umbrella to the ground. A few seconds later, lightning struck. Apparently, the umbrellas had worked as a kind of antenna, drawing in at least a portion of the huge voltage field associated with lightning. Fortunately, there was no direct hit, so the amount of energy must have been small enough to prevent injuries or serious damage.

Once recovered from the 'attack' from above one of the umbrella owners started to explore the subject of thunderstorms and translated his findings in a practical circuit that would allow him to get a better insight into the 'workings' of this fascinating natural event. At this point we should hasten to add that the design discussed here is intended **for lightning detection only**. Unfortunately, this type of detection does not tell you the distance between the lightning and the detector, and neither does it provide an early warning of an imminent lightning bolt. On the other hand, the circuit does provide a useful indication of the electrical field strength that comes with a lightning discharge.

Thunderstorms

While scientists hold different theories about the exact origins and effects of thunderstorms, they generally agree about the following.

A large voltage difference arises between a thundercloud and the earth's surface, resulting in an extremely strong electrical field. Eventually, the field strength between the cloud and the earth rises to a level where, as a result of ionisation, a region of gas (air) is turned into plasma. Plasma provides a conducting path for an arc discharge. Once the plasma path reaches the earth, a charge-cancelling current can start to flow between cloud and the earth. The resulting currents in the plasma path are immense.

Because the plasma path represents a certain resistance (albeit very small), heat is produced. Scientific measurements have indicated that temperatures inside lightning bolt may exceed those at the sun's surface! The sudden temperature rise causes surrounding air to expand and produce a propagating shockwave which we can hear as a thunderclap.

The shockwave, by the way, can be dangerous, and not only to your eardrums! The shorter the discharge, the more intense the shockwave. Cases have been recorded of houses being crumbled by shockwaves produced by lightning discharges.

Measuring

So how can we measure if a lightning discharge occurred in the neighbourhood? While the plasma path is being formed, the electrical potential of the plasma will be about equal to that of the thundercloud. An electrical field may be described in terms of potential difference per unit of distance (volt per metre). With the plasma path rapidly approaching the earth, the field density between it and the earth will increase.

This sudden increase in electrical field density can be detected using an antenna. Because the voltages we're talking about are massive, they need to be reduced to levels at which semiconductors are happy to work for us.

Circuit description

The practical circuit of the lightning intensity detector is show in **Figure 1**. The pickup device is a common or garden telescopic antenna with a length of about 1 m, connected to the input terminal marked 'ANT''. If the thunderstorm is still relatively far off, or if you want to have a go at recording 'summer lighting', the sensitivity of the detector may be increased by connecting the terminal identified with the earth symbol to the plumbing or central heating. The voltage picked p by the antenna is lowered by two potential dividers, R2-R3 and R4-R5.

The voltage reduction is considerable.

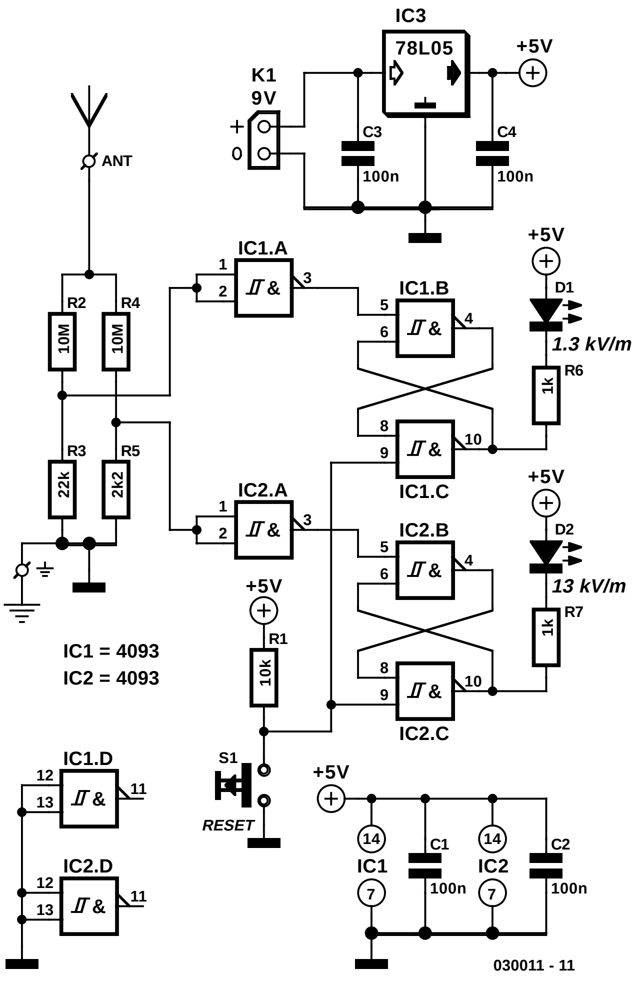
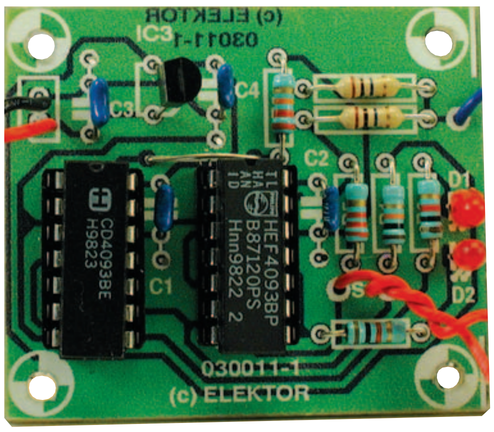


Figure 1. The core of the circuit are two potential dividers and associated flip-flops. LEDs are used to indicate and record the occurrence of two levels of electrical field strength, 1.3 kV/m and 13 kV/m.



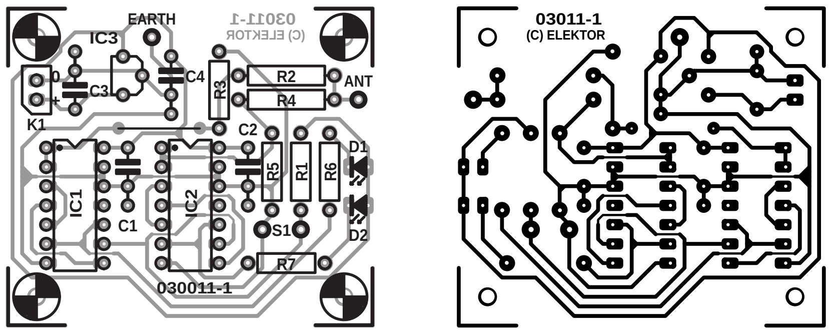


Figure 2. Copper track layout and component mounting plan of the PCB designed for the Lightning Intensity Detector (PCB available from The PCBShop).

COMPONENTS LIST

Resistors:
R1 = 10kΩ
R2,R4 = 10MΩ
R3 = 22kΩ
R5 = 2kΩ2
R6,R 7= 1kΩ

Capacitors:
C1-C4 = 100nF

Semiconductors:
D1,D2 = LED, red, low-current
IC1,IC2 = 4093
IC3 = 78L05 (see text)

Miscellaneous:
ANT = antenna, telescopic whip or 1 metre wire
K1 = 9V battery with clip-on leads
S1 = pushbutton, 1 make contact
PCB, available from The PCBShop

Potential divider R2-R3 reduces the voltage by a factor of

(R2+R3) / R3 = 214

while R4-R5 is dimensioned for a factor of

(R4+R5) / R5 = 4,546.

Each of the reduced voltages is applied to the 'set' input of a set/reset (S/R) flip-flop. Here, the flip-flops are built from NAND gates with a Schmitt trigger input. Two ICs type 4093 (IC1 and IC2) are applied in this circuit. They were chosen because Schmitt trigger gates do not allow their output to change state (i.e., High to Low or Low to High) until the voltage at the input exceeds or drops below an accurately defined upper or lower level, respectively. The hysteresis created in this way ensures the absence of an 'undefined' range between logic Low and High levels.

As soon as the antenna voltage exceeds 1.3 kilovolts (1,300 volts!), R3 will drop about 2.9 volts. That is the typical switching threshold for a 4093 operated at a supply voltage of 5 volts. Consequently, inverter gate IC1a will drop its output to logic Low and so cause the flip-flop built around IC1b and IC1c to be set, as well as LED D1 to light. Because of the latching effect of the flip-flop (a rudimentary memory device), the LED will remain on even if the antenna voltage disappears again. The LED can only be switched off by resetting the flip-flop and that is done by pressing push-button S1.

The same principle of operation applies to IC2 and LED D2, albeit that the antenna voltage has to exceed 13 kV before a voltage of 2.9 V can occur at the input of IC2a. In other words, this second detector requires a stronger electrical field to make the LED light, so it is less 'sensitive' than the other detector.

The values of R3 and R5 are, of course, subject to experimenting to see what sort of electrical fields occur around your home during an average thunderstorm.

After a lightning discharge, the circuit can be prepared for the next measurement by pressing reset button S1.

Power supply

The power supply for the circuit could not be simpler. A 9-V (PP3 or 6F22) battery supplies a 78L05 regulator (IC3), which in turn provides a stable 5-V supply rail for the rest of the circuit. If desired, the circuit can be made to work from a 6-volt battery pack by using a low-drop regulator in position IC3. For example, the 2951 requires just 5.5 V at its input to supply a stable 5-V output voltage.

The current consumption of the circuit is modest at just a few milliamps. Because the circuit does not have to be on all the time, a PP3 battery will probably last for a number of years.

Construction

A printed circuit board has been designed for the detector (**Figure 2**). You can either etch it yourself or order one ready-made from The PCBShop (accessible via the *Elektor Electronics* website). The PCB has the size of a matchbox and is unlikely to take more than half a hour or so to populate.

Small enclosures to hold the stuffed PCB and the battery should be widely available in various shapes and sizes. The antenna may be a genuine telescopic whip, but if you need to cut costs then a piece of wire with a length of 1 m will also suffice.

Warning

Direct lightning and some secondary discharges represent lethal voltages and current levels. Never use this instrument at a position above the highest point of a lighting arrestor system, in the direct vicinity of such a system, or in any other way with a view to attracting lightning.

(030011-1)

Electronic Knotted Handkerchief

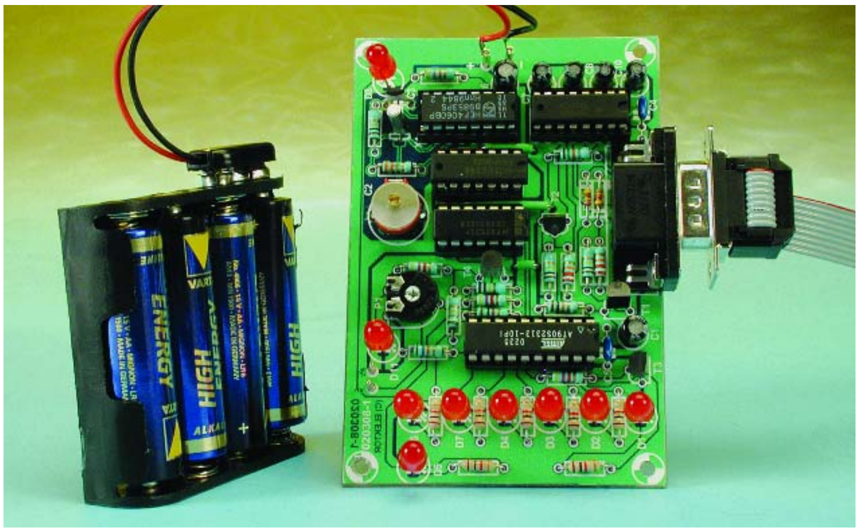
A microcontroller-based notepad

Design by J. Böcker

How many times has the birthday of that rich uncle or that six-monthly appointment at the dentist slipped your mind? How many times have you forgotten to put out the rubbish? The solution is either a little mental training — or the idea presented here: an electronic knotted handkerchief.

Key features

– Space to store from 48 to 94 appointments (depending on the number of distinct events)
– Schedule appointments up to 22 months ahead
– Set times in steps of one hour
– Advance notice period settable from 0 to 255 hours in steps of one hour
– Appointments displayed for a period from 0 to 124 hours (without advance notice), settable in units of four hours
– Battery life: one to two years using four alkaline AA cells (depending on duration and frequency of displayed appointments)



This 'electronic knotted handkerchief' is capable of remembering up to six types of appointments. When an appointment approaches, it gives a warning in the form of one of the LEDs (D1 to D4, D6 or D7) flashing. When the appointment is very near, another LED (D5) also flashes. If the capacity for more than six significant events is wanted, then up to 63 different LED patterns can be programmed. Of course, then you will have to keep track of which pattern corresponds to which event, or else your rich uncle will get your tax return and the Inland Revenue will get the flowers!

If no imminent event is indicated, LED D8 flashes every 4 s to show that the device is operating. If the battery is nearly exhausted, D8 blinks once every second.

Programming the device takes place using PC-based software which records the event data and corresponding times and downloads them into the microcontroller. After programming, the application starts up and the connection with the PC can be severed. One disadvantage should be pointed out: when the battery is changed the microcontroller will forget the time, since the clock is implemented in software using a small amount of RAM. The time must therefore be programmed again from the PC.

An end to amnesia

In order to keep the circuit of **Figure 1** as simple as possible, a facility for in-circuit programming has not been provided. The program code (not the appointment data) must be loaded into the processor using an external programming device.

The processor clock is generated using oscillator/divider IC2 along with X1. Here the selection of the particular device is important, as the characteristics vary significantly from manufacturer to manufacturer. If, despite adjustment of C2, the oscillator will not start up, the value of C3 should be reduced slightly.

Since most of the work is done in software, the circuit is relatively easy to construct. The central component is the AVR microcontroller type AT90S2313P from Atmel, which has the following features:

- 20-pin DIL package
- 1K x 16 flash program memory
- 128 bytes RAM
- 128 bytes EEPROM
- 2 timers, one configurable for PWM generation
- integrated comparator
- integrated UART with dedicated baud rate generator

IC2 is not connected directly to the power supply, but rather via R23. This is for the following reason: at low clock frequencies the current consumption of the IC is unreasonably high, probably because during switching both transistors in an output stage briefly conduct at the same time, creating a short circuit. The effect of this on the overall current consumption would be significant if R23 did not limit this current to something under 100 µA. Unfortunately this means that the output signals have much slower rise and fall times, and so they must be squared up using a Schmitt trigger (IC4). The crystal frequency of 32,768 Hz is divided by 2^{13}=8,192 in IC2 and then twice further divided by two in IC3. The result of this is (on pin 1 of IC3) a clock at 1 Hz, which is connected to the INT0 pin of the microcontroller to generate an interrupt signal.

The interrupt service routine sets PD1 high in order to set flip-flop IC3.A. As long as the processor is in the interrupt service routine the low level at the interrupt input is ignored: there is no 'interrupt within an interrupt'. R16 pulls the set signal to a defined low level when the port output is (for reasons of power consumption) set to a high impedance state.

The anodes of LEDs D1 to D7 are driven via switch T3 from the PWM output PB3/OCI of the microcontroller.

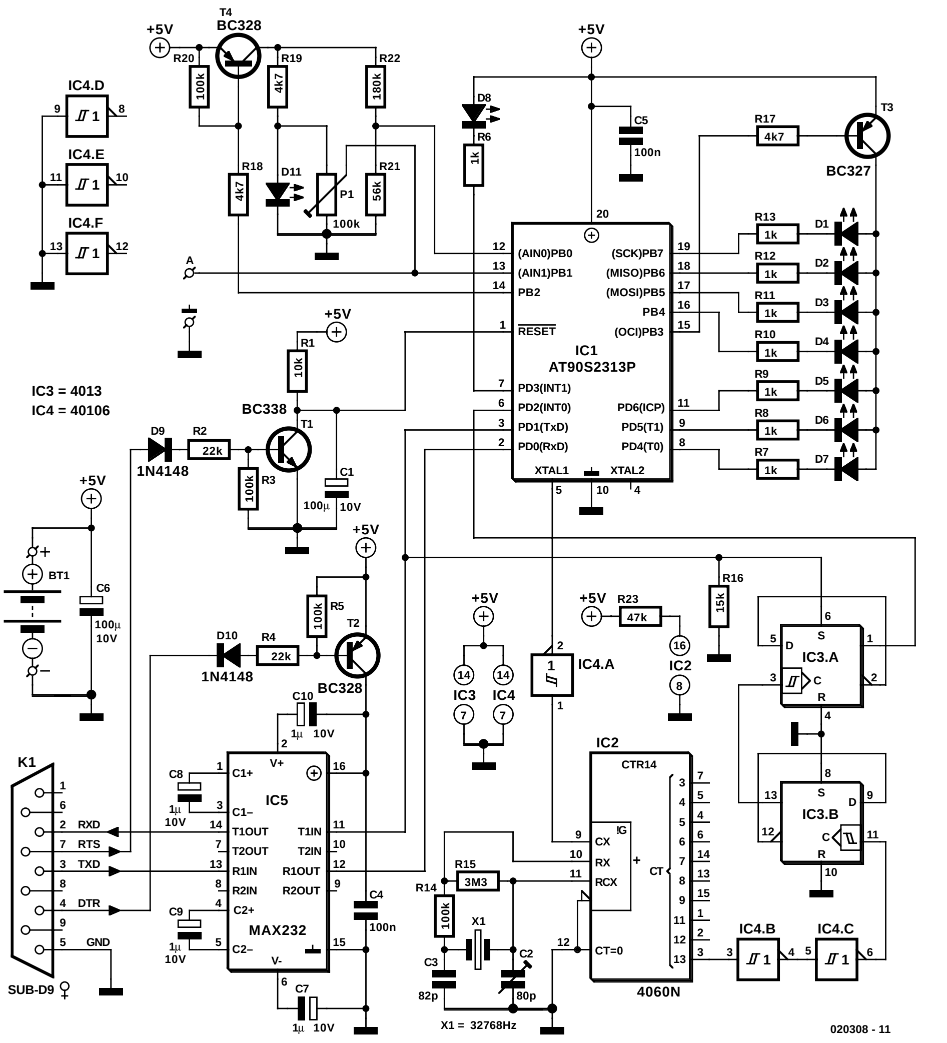


Figure 1. The serial port and battery low detector are switched off when they are not in use.

The battery voltage is measured using the internal comparator via pins PB0 and PB1. This is done just once a day, and even then only when no event (or approaching event) is indicated. If the battery voltage falls below 4 V, a signal appears on port PD7, which is not brought out of the microcontroller.

Lastly we come to the serial interface which uses the well-known MAX232 level shifter (IC5). PD1 drives the TxD signal (also disabling interrupts). Also, the PC can reset the microcontroller using the RTS signal on the serial interface. Without a PC connected T1 is turned off and so a logic high appears at the reset input. If RTS is at logic zero (+12 V), T1 conducts and pulls the reset input low (active). If RTS is at logic one (–12 V), D1 does not conduct and so neither does the transistor. Then C1, which forms part of the power-on reset network with R1, starts to charge and then the program runs.

Power saving

Power-saving measures are essential in a battery-operated circuit that is to run for a year or more. In order to achieve this goal, the LEDs are driven using a PWM signal that can be configured by programming from the PC. This can significantly reduce the current consumption from as much as 4 mA per LED, if the circuit is not to be used in bright ambient lighting. The power-on indicator is driven with a duty cycle of less than 1 % (1/32 s every 4 s); this gives a clearly visible flash.

The FETs in modern components behave – as far as current consumption is concerned – like capacitors. As the operating frequency increases, so does the current consumption. The microcontroller draws around 5 mA in static operation. At a clock frequency in the MHz range this increases by an enormous factor. Here a 32768 Hz watch crystal is used, which is slow, readily available and inexpensive. Also, it is easy to derive a 1 Hz clock by dividing the frequency by 2^{15}.

Since AVR microcontrollers execute practically all instructions in one clock cycle (and are therefore twelve times faster than an 8052), the processor carries out over 30,000 instructions per second. This is more than enough for this application.

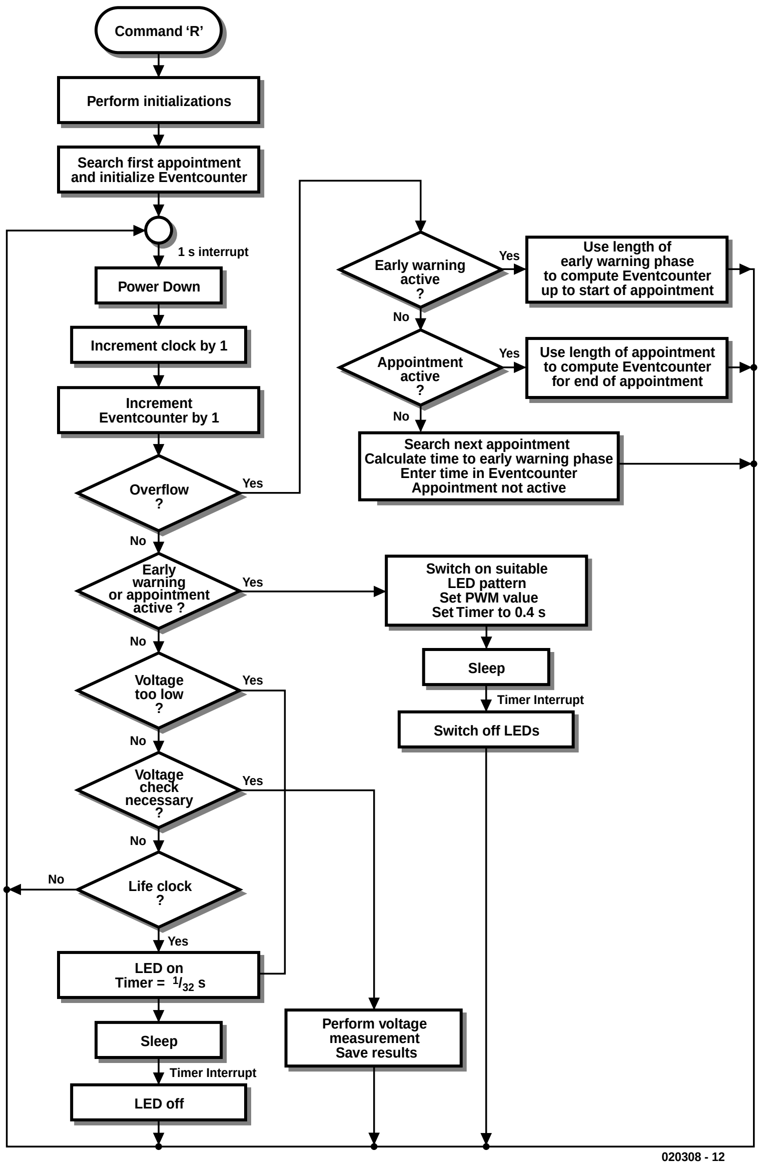
Since the baud rate for the microcontroller's integrated UART is derived from the processor clock, communication with the PC is unfortunately only possible at 110 baud. Since the amount of data to be transferred is not enormous, the slow transfer speed does not present a great problem.

Even when the microcontroller draws only 5 mA, that is still too much. Batteries would need to be changed at least once a month, just to keep the microcontroller running; and that doesn't even count the power used by the LEDs and interface chips. An important step in saving power is nevertheless to switch off all unnecessary current-consuming devices, namely the battery voltage measurement circuit (using T4) and the serial interface circuit (using T2).

Of course we cannot switch off the microcontroller in this way, but Atmel does offer several power-saving modes. In sleep mode the microcontroller core is put into a quiescent state, and the device's current consumption falls to 1 mA. However, this is still too much.

In power-down mode, on the other hand, the timers and oscillator are shut down in addition to the processor core. The registers and RAM, however, retain their contents. The microcontroller can only be woken up from this mode by an external interrupt or reset. On interrupt, the processor starts executing code from the address where it was put into the power-saving mode.

Furthermore, the internal components of the CPU (driver stages for the port outputs, the integrated comparator, the timers along with the compare unit used for PWM generation) can be turned off by setting or clearing dedicated control bits in the I/O registers.



Command 'R'
Perform initializations
Search first appointment and initialize Eventcounter
1 s interrupt
Power Down
Increment clock by 1
Increment Eventcounter by 1
Overflow ?
Yes
No
Early warning or appointment active ?
Voltage too low ?
Voltage check necessary ?
Life clock ?
LED on Timer = 1/32 s
Sleep
Timer Interrupt
LED off
Early warning active ?
Use length of early warning phase to compute Eventcounter up to start of appointment
Appointment active ?
Use length of appointment to compute Eventcounter for end of appointment
Search next appointment Calculate time to early warning phase Enter time in Eventcounter Appointment not active
Switch on suitable LED pattern Set PWM value Set Timer to 0.4 s
Switch off LEDs
Perform voltage measurement Save results
020308 - 12

Figure 2. Flowchart for the operating state.

Software: part 1

The microcontroller program, which is written entirely in assembler code, operates in two different states: the programming/configuration state and the operating state. In the programming state the microcontroller communicates with the PC over the serial interface.

Immediately after a reset the microcontroller emits the sequence of characters 'HELLO' and then waits for a command to arrive. As an extra check that reset has occurred and that the microcontroller is operating correctly, all the LEDs flash.

The available commands which can be sent to the microcontroller are listed in **Table 1**. A few of these commands are for development and debugging; the majority are either for programming events or for testing the hardware.

If the 'R' command is sent, all the LEDs flash once and the controller switches into operating state.

Software: part 2

The operating state is configured to use an absolute minimum of power. The serial interface is no longer checked. Once a second the microcontroller is woken up from the power-down mode by an interrupt, does whatever work is necessary and then returns as quickly as possible to the power-down mode.

Essentially, registers R0 to R3 are used to count seconds from a virtual 'time zero'. All event times are reckoned relative to this zero reference point. When the PC puts the unit into operating mode it stores the time difference between this reference point and the actual time in registers R0 to R3.

Registers R4 to R7 are used to store the number of seconds until the next appointment, or until some other state change occurs, such as at the beginning of the advance warn-

Table I. Command/function overview

Command	Function
s	Enter idle mode
p	Enter power-down mode
e	Set LED pattern brightness to full
a	Turn all LEDs off
N	Turn serial interface off
R	Start program
B	Display flashing LED pattern, Values on stack are: LED pattern; PWM setting; on period in units of 128 s
C	Turn supply to battery low detector on
c	Turn supply to battery low detector off
m	Run continuous battery low detector test and output results continuously over serial interface
v	Push next value on stack
d	Dump virtual memory (EEPROM and RAM)
w	Write data stream into virtual memory. Values on stack are: count; start address

ing period for an event. On each one-second interrupt the value in R4 to R7 is decremented by one. When the value reaches zero, the appropriate action to be taken must be determined. The corresponding state is set and relevant program code segment is executed.

Immediately on entering the operational state, or when an appointment expires, data for the appointment is read from bit-addressable memory space and loaded into registers R4 to R10, R13 and R14.

Registers R11 and R12 form a counter used in the voltage measurement process. This counter is initialised with the hexadecimal value FEAD and incremented every 256 seconds. When the counter rolls over (roughly every 24 hours) a voltage measurement is carried out.

Once the last appointment has passed interrupts are disabled and the unit goes into the power-down mode permanently. In this case the batteries should be removed, since the oscillator is still running and consuming energy.

The flowchart in **Figure 2** shows the general sequence of operations in the operating state.

Software: part 3

The electronic knotted handkerchief is programmed using a PC. This of course requires further software, this time running on the PC. The type of event and the time are recorded, the information is downloaded into the device and the timer started. It is also possible to carry out hardware tests and configure various aspects of the circuit.

The program was written in Java using the free personal version of Borland JBuilder. Although Java might not seem the most obvious choice for this application, it is in fact an eminently suitable language. Unfortunately the program takes rather a long time to start up, and the 'Swing' class responsible for the user interface will run sluggishly on older PCs. The Java Communication API (CommAPI), which can be downloaded from Sun's website, is used to drive the serial port.

It is not possible to just run a Java program directly on a PC. First version 1.3 or 1.4.1 of the Java Runtime Environment (JRE) from Sun must be installed. This is available for free download at
http://java.sun.com/getjava/download.html.

Older versions will not work with this program and should not be used. The Microsoft Java VM supplied with older versions of Windows will not work either.

The program ELTAKO.EXE and the DLL for the CommAPI win32com.dll can be copied to any desired directory (although both files should go in the same directory). In order to install the CommAPI class, the file javax.comm.properties should be copied into the directory JRE\lib, or into whatever directory the RDK was installed. When you open a DOS window and enter the command

```
java -verbose | more
```

you should see the correct path to the RDK. If this does not work, you can always search for the JRE directory.

The program is started by clicking on ELTAKO.EXE.

When the button labelled 'NEW' is clicked, a new event is created in the left-hand area. You can use 'DEL' to delete an event and all the appointments associated with it. When you click on an event, it is displayed in the middle of the window. Here you can now enter the name of the event, the LED pattern which is to correspond to it, the amount of advance notice required and the duration. The duration must be a multiple of 4 hours. In the right-hand area the times for the event are shown. These must be in chronological order. To delete an appointment, the date can be blanked using the backspace key. This may create gaps in the appointment list: press the 'TIDY UP' button, and the times are sorted and deleted entries removed.

Once all the appointments have been entered, they can be saved using the 'SAVE' menu item; you can reload them at a later date using 'LOAD'. Next, select the 'DOWNLOAD' menu item. A new window appears and the data are converted into bit-addressable format. Error messages, along with details about their cause, will appear at the top of the window. Errors should be corrected in the main window before attempting to download again.

If the status line shows 'Data translation OK', you can proceed to connect the device to the PC and press the 'TRANSMIT' button. After a reset (the LEDs flash) the data are transferred and the time counter is set to the correct value. Then the unit is switched to its operating state; it can now be disconnected from the PC. Make sure that the PC's clock is set correctly.

Construction and testing

The circuit board shown in **Figure 3** is rather sparsely populated, and so construction should present no difficulties. The four wire links (one is under the microcontroller) must of course not be overlooked. For this reason the microcontroller should be fitted in a socket. All the other ICs may be fitted in sockets, although this is not compulsory.

The power supply takes the form of four 1.5 V cells, which are most conveniently fitted in the battery compartment of a suitable enclosure. Alternatively a suitable battery holder can be glued into the enclosure.

COMPONENTS LIST

Resistors:
R1 = 10kΩ
R2,R4 = 22kΩ
R3,R5,R14,R20 = 100kΩ
R6-R13 = 1kΩ
R15 = 3MΩ3
R16 = 15kΩ
R17,R18,R19 = 4kΩ7
R21 = 56kΩ
R22 = 180kΩ
R23 = 47kΩ
P1 = 100kΩ preset

Capacitors:
C1,C6 = 100µF 10V radial
C2 = 80pF trimmer
C3 = 82pF
C4,C5 = 100nF
C7-C10 = 1µF 10V radial

Semiconductors:
D1-D8,D11 = LED, red, low current
D9,D10 = 1N4148
IC1 = AT90S2313-10PC, programmed, order code **020308-41**
IC2 = 4060N
IC3 = 4013
IC4 = 40106
IC5 = MAX232
T1 = BC338
T2,T4 = BC328
T3 = BC327

Miscellaneous:
BT1 = 4 AA or AAA batteries with holder
K1 = 9-way sub-D socket (female), PCB mount, angled pins
X1 = 32.768kHz quartz crystal
Disk, order code **020308-11** or Free Download

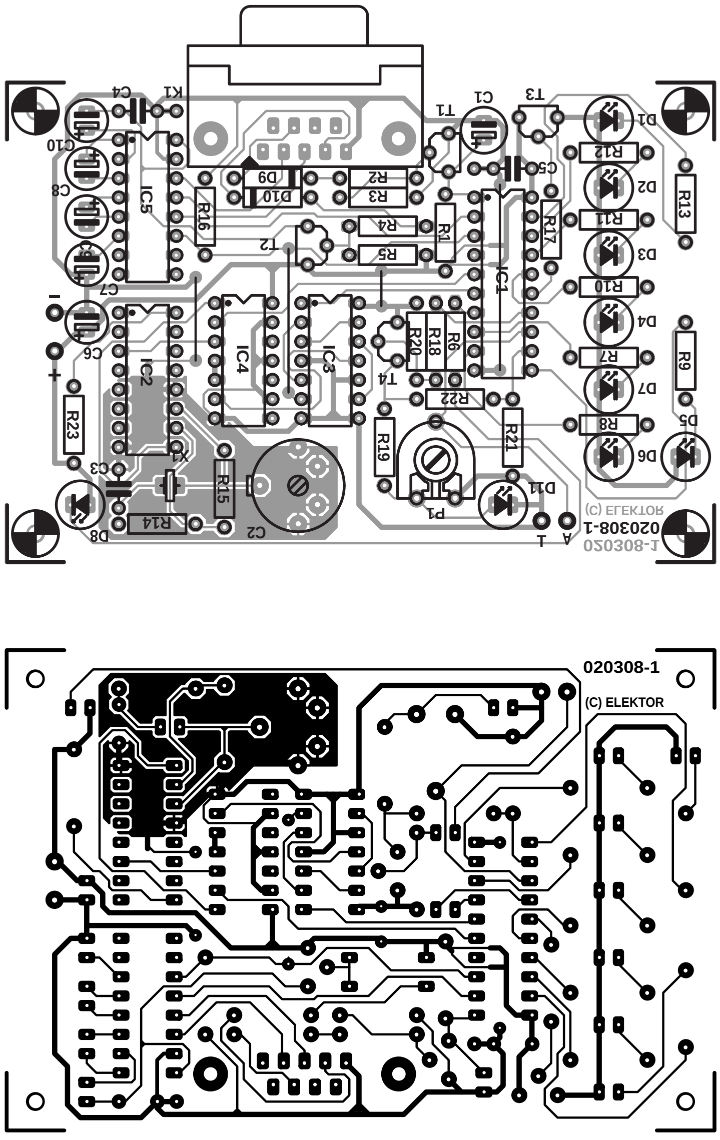


Figure 3. Everything fits on one printed circuit board.

First IC2 and IC4 should be fitted in their sockets and the device switched on. C2 should be adjusted until the oscillator operates reliably. The current consumption of the circuit should be no higher than 30 µA with these components fitted.

Before fitting the microcontroller into its socket it should be programmed from the file MAIN.HEX. You can do this yourself (the software is available from the *Elektor Electronics* website or on diskette number **020308-11**) or a ready-programmed microcontroller can be obtained (product number **020308-41**).

Before fitting the unit into an enclosure it is worth going through the hardware test and configuration procedure. For this, the electronic knotted handkerchief should be connected to the serial port of a PC using a straight-through RS232 cable with a male connector at one end and a female connector at the other. The program HW-TEST should then be run. Press the RESET button and a connection will be established. You can now test operation by turning all the LEDs on and off individually. The slider allows you to set the LED brightness, and when 'OK' is clicked the value is stored permanently.

The final step is to calibrate the battery low detector. First activate the circuit using the 'POWER ON' button and then adjust P1 for a voltage of 0.9 V between test point A and ground. Once the tests have been completed the connection can be broken using the 'UNPLUG' button.

The timing accuracy of the unit leaves a little to be desired. This is because normal crystals have a relatively large tolerance relative to their nominal frequency, even though according to the data sheet the maximum error in the crystal frequency should be only 10 ppm. In practice errors of more than 3 s per month are found.

(020308-1)

Free Downloads

– PC and microcontroller software.
File number: 020308-11.zip
– PCB layout in PDF format.
File number: 020308-1.zip
www.elektor-electronics.co.uk/dl/dl.htm, select month of publication.