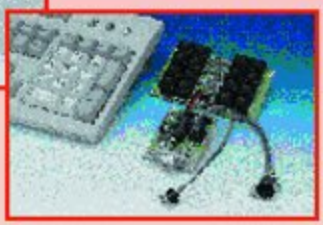
THE ELECTRONICS & COMPUTER MAGAZINE
APRIL 2000
£2-95
www.elektor-electronics.co.uk

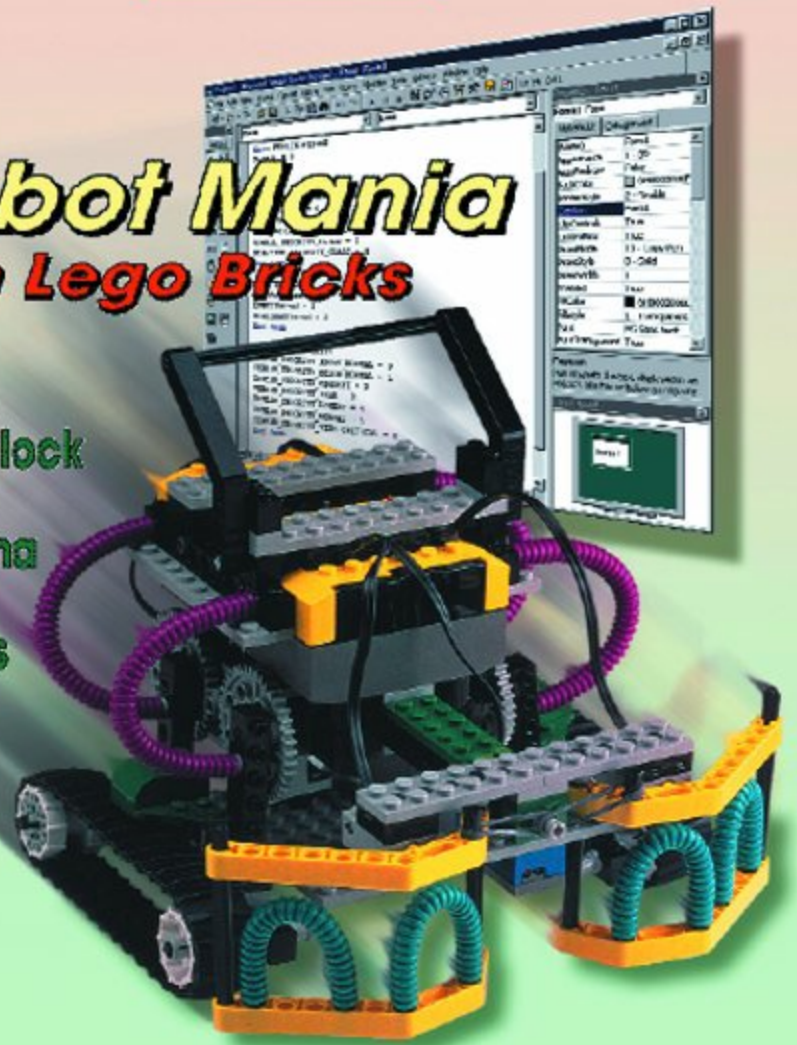
ELEKTOR ELECTRONICS

Low-Impact Muscle Stimulator



HotKeys: PC Power Typing

Robot Mania with Lego Bricks



Multifunctional Clock

Active Whip Antenna

Vibrating Batteries for GSM Phones

9 770268 451061 04

L.I.M.S.

Low-Impact Muscle Stimulator

with treatment programs

Design by K. Walraven

This is a DIY version of a device that can be found in various forms in the clinics of physiotherapists and massage institutes. It produces a series of electrical shocks that stimulate the muscles. The intensity and duration of the shocks are adjustable, and we have given great attention to the safety of this device.

Use this equipment only with battery power.
Never use this equipment in the vicinity of the heart! It should preferably be used only on the arms and legs.
Always set the amplitude control to the minimum level when starting a treatment.

Anyone who has ever visited the clinic of a physiotherapist or a sports masseur will have been amazed at the large amount of complex-looking equipment to be found there. In this area, as in many others, modern technology (and in particular electronics technology) has become indispensable in a relatively short time.

One type of equipment that is most often used by these people is a muscle stimulator. There are all sorts of different models available with a variety of names, but they all work in essentially the same manner. Two electrodes are attached to some part of the body a small ways apart, and a pulsating current is passed between the electrodes. The objective of this is to stimulate the muscle located between the electrodes.

This usually starts with loosening up a cramped muscle and promoting the blood circulation in the muscle tissue. A

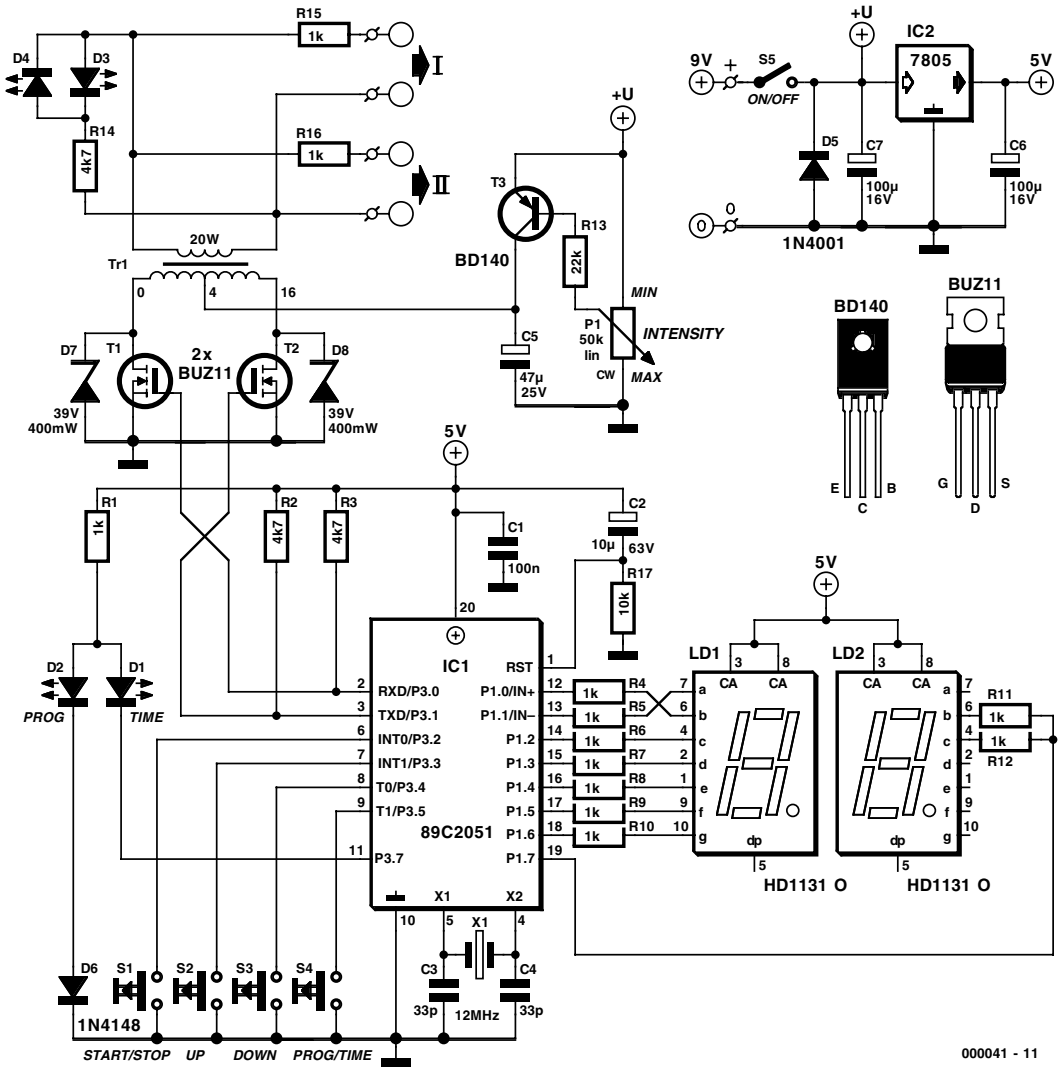


Figure 1. The use of a pre-programmed microcontroller allows the hardware to be kept to a minimum.

low-intensity pulsating current is adequate for this. With some therapeutic treatments, the only objective is to create a general sense of wellbeing, which can be achieved using even weaker shocks. If on the other hand the equipment is used to strengthen the muscles, the pulses that are used are significantly stronger.

However, we don't want say too much about this, since you might otherwise get the false impression that we are experts on the subject. Just to avoid any misunderstanding, let us say that this is absolutely not the case. All that we are doing is making the necessary equipment available, in response to repeated requests. From what we have been able to find out, the technology used in this sort of equipment is in fact fairly simple. We have translated this technology into a DIY design that is as safe and reliable as possible. That is our job and our skill.

We would rather not say anything about how to use the equipment. Instead, we strongly recommend that you consult a doctor or other specialist. You definitely should not use the muscle stimulator if you have not first informed yourself about what can and cannot be achieved with electrical muscle stimulation. Furthermore, do not regard this as a kind of toy. Even though it is powered by a battery, it can without question produce very intense current pulses, and these could have undesirable effects if the equipment is not used properly. Depending on the skin resistance, the peak voltage across the electrodes is 35 to 45 V; with no load it can easily reach 50 V!

If all that you are interested in is loosening up certain muscles or enjoying a relaxing massage, we would actually advise you to leave this electronic stimulator alone and instead ask you spouse or another family member to help you. After all, there's nothing wrong with a normal hand massage! Then at least you do not have to worry about possible harmful effects.

Design

The complete schematic diagram of the stimulator is shown in Figure 1. We can distinguish between two things in terms of its operation: the generation of the electrical pulses and the implementation of a few special functions.

The first of these is remarkably simple. Microprocessor IC1 (an 89C2051) is programmed to generate pulses on its RxD and TxD outputs after the Start/Stop button S1 has been pressed. These pulses drive MOSFETs T1 and T2 alternately into saturation. Transformer Tr1 boosts the voltage of these pulses by a factor of around 10, and they are then applied to the electrodes I and II. That's all there is to it. The strength (amplitude) of the pulses

is controlled by simple current source formed by T3, which can be adjusted using P1. The current source charges electrolytic capacitor C5, and the voltage on this capacitor is used for the output pulses. This approach also ensures that only a certain limited amount of energy reaches the transformer for each time interval, which is very important with regard to safety. LEDs D3 and D4 act as a simple amplitude indicator for the output pulses.

This takes care of the pulse generation. Now you may be wondering why we have used a microprocessor for the pulse generator, since this could be done just as well with a couple of transistors and a 555 timer IC. The presence of the processor is entirely due to a number of extra functions that have been included in the stimulator. These extra functions also explain the presence of the two-digit display (LD1 and LD2), the two indicator LEDs D1 and D2 and the control buttons S1 through S4.

The buttons can be used to select a specific *treatment time* and a particular *treatment program*. You will see shortly how this is done. To avoid any misunderstanding, we should say right away that output pulses can be produced only after both a time and a program have been selected. Until this has taken place, pressing the start/stop button S1 has no effect. This has been done for safety reasons, in order to prevent the stimulator from unexpectedly delivering a continuous current at an unguarded moment.

Before we go any further into the details of the program options of the stimulator, we must first divert our attention to a more detailed description of the pulse signals that it produces.

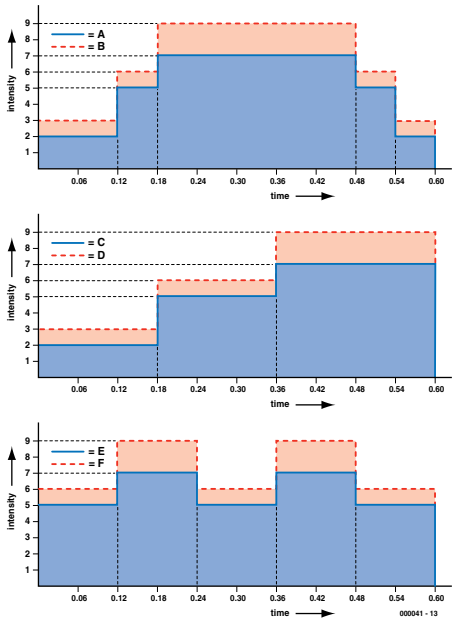


Figure 3. There is a choice of three different arrangements for the automatic treatment programs, each with two intensity levels.

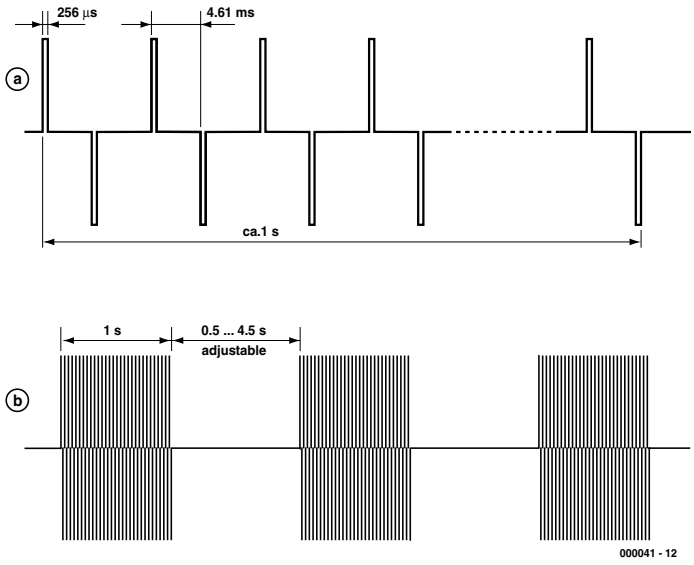


Figure 2. The pulse signal produced by the muscle stimulator.

Pulses

After immersing ourselves deeply into the available literature, we came to the clear conclusion that the developers of such devices are remarkably unanimous with regard to the width and repetition rate of the pulses. Based on this, we have decided on alternating positive and negative pulses with a width of 256 µs and a repetition rate of 109 Hz. This corresponds to an idle time of 4.61 ms.

The pulses do not appear continuously at the output, but rather in packets or bursts with a duration of around one second. A typical burst is illustrated in **Figure 2**. This naturally does not show all the pulses in the burst, since we'd need a lot bigger page for that.

The interval between successive bursts can be set between 0.5 and 4.5 seconds. The resulting signal on the electrodes appears as shown in **Figure 2b**. The intensity of the shocks depends almost as much on the repetition interval as on the pulse amplitude set by P1. The various treatment programs take advantage of this fact.

Program and time

As already noted, two things must selected before the circuit can start delivering pulses: the treatment time and the treatment program.

The time is set by selecting the 'time' mode using button S4. The 'time' indicator LED D1 is on when the unit is in the 'time' mode. After this, the time can be set between 1 and 19 minutes using the up and down buttons S2 and S3. The time is shown on the display. The pulses will stop after the selected time has expired.

If the unit is switched to the 'program' mode by pressing button S4, the indicator LED D2 lights up. In this mode, one of a total of fifteen programs can be selected. The programs labelled 1 thorough 9 are the simplest ones, since they work with a constant intensity level based on selecting the burst interval time. When the display shows a '9', the intensity is at maximum, since the minimum burst interval of 0.5 s is used. Program 8 corresponds to an interval time of 1 s, '7' represents an interval time of 1.5 s and so on. This 'illogical' reverse sequence has been chosen intentionally so that the strongest program has the highest number.

The programs indicated by 'A' through 'F' are more sophisticated, since various intensity levels are used during the course of the treatment session. There are actually only three different programs, each in a weak and a strong version.

Figure 3 shows how programs A through F are set up. The intensity is divided into the same levels as for programs 1 through 9. This means that the burst interval is the shortest, and the intensity the greatest, at level 9. The horizontal axis shows the treatment time — which as you know can vary.

In programs A and B, the intensity of the pulses is increased to its maximum

Project software

Disk # 000041-1 supplied through Readers Services

LIMS		2.734	21-10-00	10:49	lims
LIMS	LST	86341	21-01-00	10:49	lims.txt
LIMS	HEX	5.881	21-01-00	10:49	lims.HEX
LIMS	PRJ	2.112	21-01-00	11:33	lims.prj
CONTENTS	TXT	0	08-02-00	11.49	contents.txt
LIMS	M51	4.200	21-01-00	10:49	lims.m51
LIMS	C	16.984	08-02-00	11:38	Lims.C
LIMS	OBJ	5.683	21-01-00	10:49	Lims.OBJ
README	TXT	220	08-02-00	11:43	Readme.txt

(Sophos data integrity and fingerprint files not listed)

value in three steps, following which it remains at this level for a reasonable length of time and then drops back down in steps. Program A is the weak version, and program B is the strong version.

In programs C and D, the treatment time is divided into three equal intervals, and the intensity is gradually increased in each of these intervals. Once again, C is the weak version and D is the strong version.

In programs E and F, the treatment time is divided into five intervals, and the intensity alternates between two levels in each of these intervals. Version F is the stronger of the two.

The durations of the individual phases of programs A through F naturally depends on the

Software

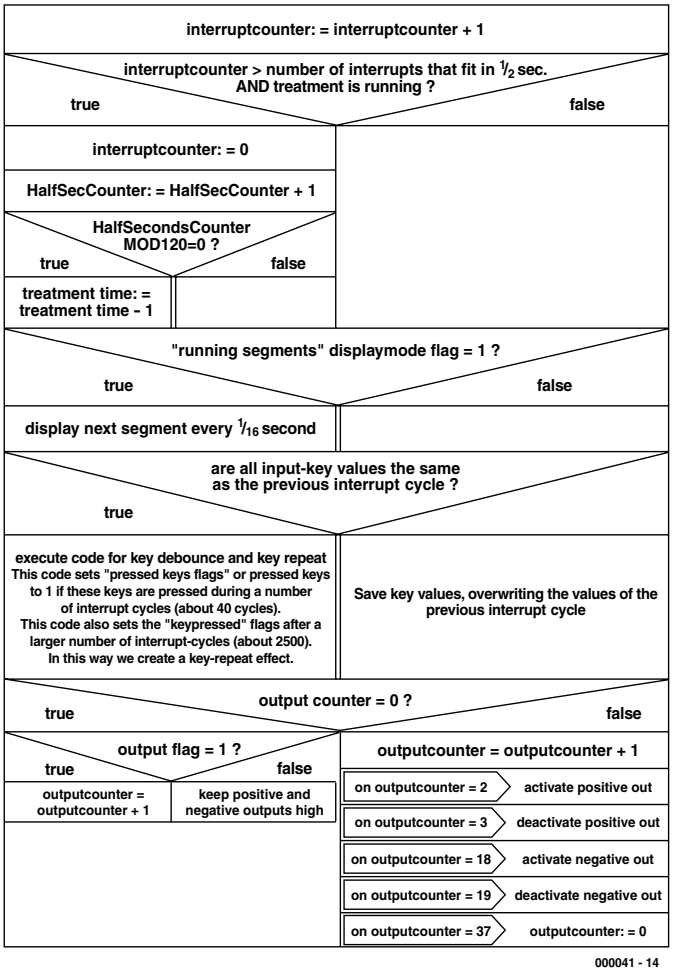
The software contained in the microprocessor consists of a main routine and an interrupt routine. The interrupt routine handles all real-time processing, such as keeping track of the time, generating the pulses and scanning the keypad (with debouncing). Pulses are generated as soon as the main routine has set a flag. In the reverse direction, the interrupt routine sets flags whenever it detects that a valid button has been pushed. The time (in the form of a counter with a resolution of half a second) is also made available to the main routine. The main routine responds to the pushbuttons, drives the display and determines when output pulses must occur.

The processor works with a 12-MHz crystal. The internal Timer 0 works in mode 3, which means that it divides by 256 (8-bit counter) and has an input frequency of (12 MHz ÷ 12) = 1 MHz. Timer 0 thus generates an interrupt every 256 μs. The number of interrupts is counted and used to control the circuit. For example, output P30 is set high when the interrupt count is 2 and it is set low again when the count is 3. Output P31 is set high when the count is 20 and again low when the count is 21.

At a count of 37, the counter resets itself (to 1) and a new cycle begins. The time between two pulses is thus 18 times 256 μs, or 4608 μs. A full cycle lasts 36 times 256 μs, which is 9216 μs. Each cycle is always completed once it is started. Timekeeping is based on counting 1964 interrupts for every half second. Here a full second is not exactly one second, but instead 1.000448 s.

The main program actually consists of a state machine. In order to prevent the accidental generation of pulses, there are two parts, one with and the other without pulses. Only after all of the prerequisites have been satisfied is it possible to transfer to the part that has pulses.

Interrupt routine of the Low Impact Muscle stimulator (LIMS)
This routine executes on every overflow of the 8 bit timer 0 (timer 0 is used in Mode 3)
The timer counts at CLOCKFREQ/12, so the interrupt routine executes every 256 µs (@12 MHz crystal)



000041 - 14

In the idle state, the software checks to see if a button is pressed. Until this happens, the program runs in an endless loop. As soon as a button is pressed, the program takes action. It switches over to the part of the program with output pulses only after a program has been selected, the treatment time has been set and the start button has been pressed.

Based on the treatment time and the selected treatment program, the main program then periodically sets a flag that causes the interrupt routine to generate pulses. When the stop button is pressed or the time has expired, the other half of the program is again called.

The 'running 8' display is an exception, in that it is generated in the interrupt routine rather than in the main program.

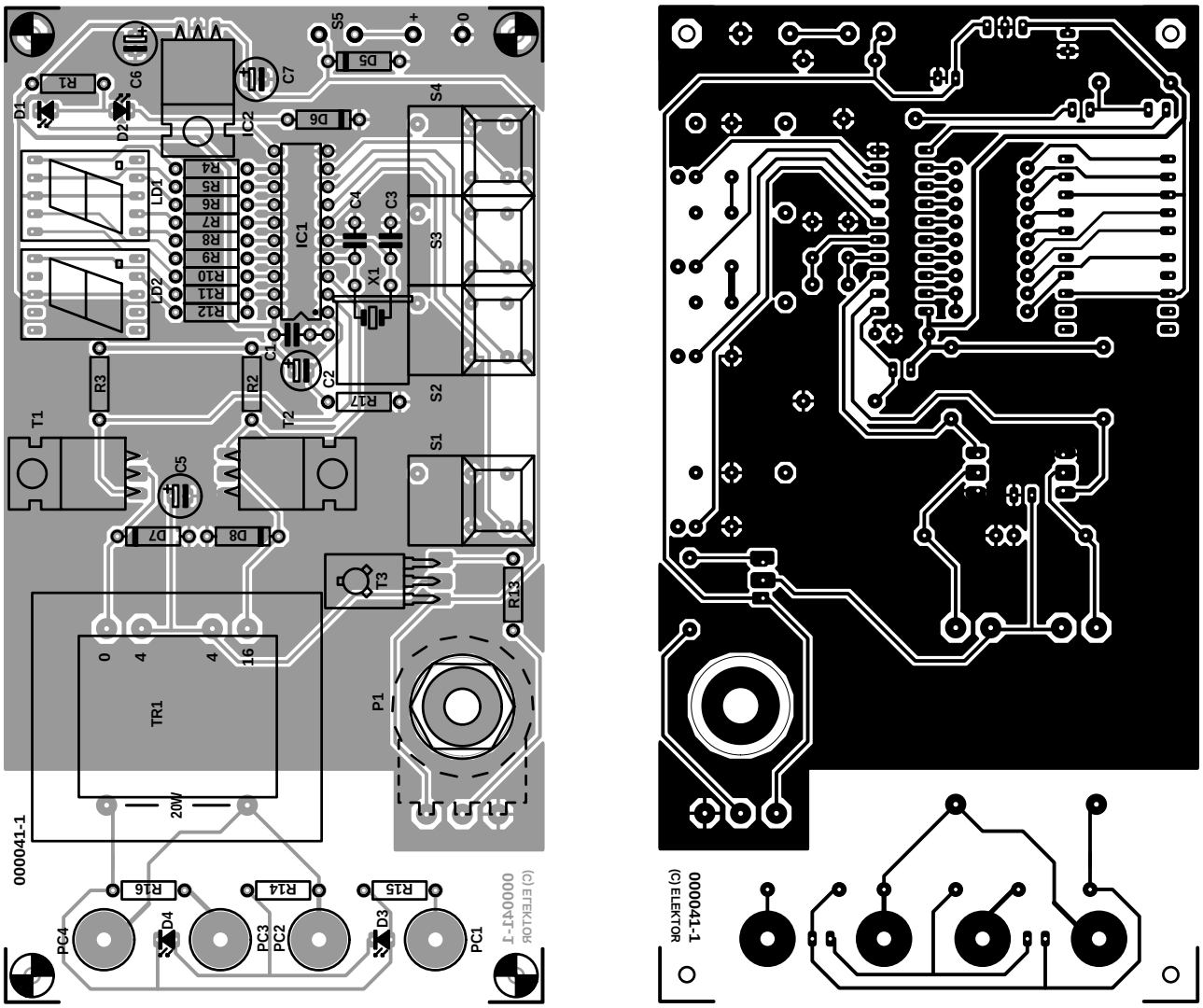


Figure 4. Aside from the processor, the transformer and the displays, there aren't a lot of components on the circuit board.

selected treatment time. The microprocessor always calculates the correct distribution.

Starting and stopping

As soon as the start/stop button S1 is pressed after the time and program have been selected, the stimulator starts to generate pulses and a 'running 8' appears on the display. As long as pulses are being generated, it is not possible to change the selections, and only the start/stop button can be used. If any of the other buttons is pressed, the display shows the number of the selected program or the remaining treatment time. The display automatically returns to showing a running '8' after about five seconds.

The program can be interrupted by pressing on the start/stop button. This amounts to a sort of pause function, since the program continues if the button is pressed again. While the program is paused, the program number or the remaining time is displayed, and it is possible to change the settings.

After the set time has expired, pulse production stops automatically and the default values are shown on the display: '–' for the program and '0' for the time. New values for both of these must be selected before a new treatment cycle can be started.

Safety and power

With a circuit such as this, safety must have the highest priority. We have thus given it the proper amount of attention.

An important point is that pulses can never be produced 'by accident' if the start/stop button is pressed at the wrong time. It is always necessary to intentionally select a program and a time before this button has any effect.

The benefit provided by the current source T3 has already been mentioned. It limits the amount of energy that is supplied to the transformer. If a voltage regulator had been used instead of a current source, the voltage would try to remain constant, with the consequence that occasionally much more current would be delivered than is actually necessary. This could lead to dangerous situations.

However, using a current source also has an unavoidable disadvantage, which is that when there are no output pulses there is no current consumption, and consequently capacitor C5 will be charged to the maximum voltage. The first pulses after this could thus have a dangerously high intensity. This is countered by having the microprocessor drive *both* transistors (T1 and T2) at the same time when

COMPONENTS LIST

Resistors:
R1,R4-R12,R15,R16 = 1kΩ
R2,R3,R14 = 4kΩ7
R13 = 22kΩ
R17 = 10kΩ
P1 = 50kΩ linear potentiometer

Capacitors:
C1 = 100nF
C2 = 10µF 63 V radial
C3,C4 = 33pF
C5 = 47µF 25V radial
C6,C7 = 100µF 16V radial

Semiconductors:
D1-D4 = high-efficiency-LED, red (3 mm)
D5 = 1N4001
D6 = 1N4148
D7,D8 = zener diode 39V/0.4W
T1,T2 = BUZ11 (BUZ10, BUZ100)
T3 = BD140
IC1 = 89C2051-12PC (programmed), order code **000041-41**
IC2 = 7805 (or LM2940T-5, 4805)

Miscellaneous:
S1-S4 = Digitast pushbutton (ITT/Schadow)
S5 = on/off switch
Tr1 = 100-V line output transformer 0-4-16 Ω/20 W (Conrad Electronics or Monacor/Monarch)
X1 = 12 MHz quartz crystal
LD1,LD2 = HD11310
Enclosure: e.g., TEKO 160x95x61 mm
PC1-PC4 = miniature wander sockets
Disk, project software (source code and hex code): order code **000041-11** (see Readers Services)

no pulses are being generated. This discharges C5, and the first pulses will actually be weaker than normal. This is a pleasant side effect, since it results in a sort of 'soft start' effect for each cycle.

The choice of the power supply is also a critical issue with regard to safety. This circuit must be powered *only* from a battery, and *absolutely not* from a mains adapter. If you use rechargeable batteries, the battery charger must *never* be connected to the stimulator while it is in use.

The power source can be either six penlight cells or a 9-volt battery, as desired. The current consumption is around 50 mA, which is not all that low. However since a treatment session lasts at most 19 minutes, even a 9-V battery will last for at least 30 sessions. If you replace the voltage regulator (IC2) with a low-drop type (LM2940-5 or 4805), the battery can be used until it is discharged to 6 V, and the number of sessions that you can get from a single battery is even greater.

Construction

The printed circuit board for the muscle stimulator is shown in **Figure 4**. Since there are actually only very few components in this circuit, we hardly need to say much about building it. If you use the components specified in the components list and mount them as shown in the layout drawing, there's not much that can go wrong.

Transformer Tr1 is what is called a '100-V' output transformer. Even though output transformers are not all that popular any more, it is reasonably easy to obtain. Both Conrad Electronics and Monacor (or Monarch in some countries) still have this type of transformer in their catalogues. We need a type that has 0/4/16-Ω outputs and can deliver 20 W. Although it's true that we do not need anything like 20 watts here, this is the only transformer that has the correct winding ratio (1:10).

After you have finished assembling the circuit board and have carefully inspected it (pay particular attention to the quality of the solder joints and watch out for possible shorts!), you can look for a suitable enclosure. The type shown in the components list is attractively compact, but it's a bit on the small side. If your transformer turns out to be somewhat larger than the one used in our prototype model, the whole assembly may not fit in this enclosure, and you will have to look for something bigger.

As indicated on the component layout drawing, a number of components (such as the transistors and the voltage regulator) should be mounted flat on the board in order to limit the height of the assembled board. The photograph of the board in **Figure 5** clearly illustrates this.

The board is mounted using four small screws that are glued to the rear side of the front panel of the enclosure. However, it's better to make the first tests without the front panel in place, since it's easier to access the board in this case. If there are any problems, it's then easier to check whether the supply voltage is present at certain key points.

We designed a front panel for our prototype that matches the enclosure shown in the components list, as shown in **Figure 6**. Of course, you are free to choose something different if you want to.

Electrodes

In our experiments, we were able to successfully use homemade electrodes consisting of small pieces of unetched printed-circuit board material around 2 by 3 cm in size. These were clamped to an arm or leg using elastics bands. To make a good contact, place a small piece of paper that is generously moistened with ordinary tap water between

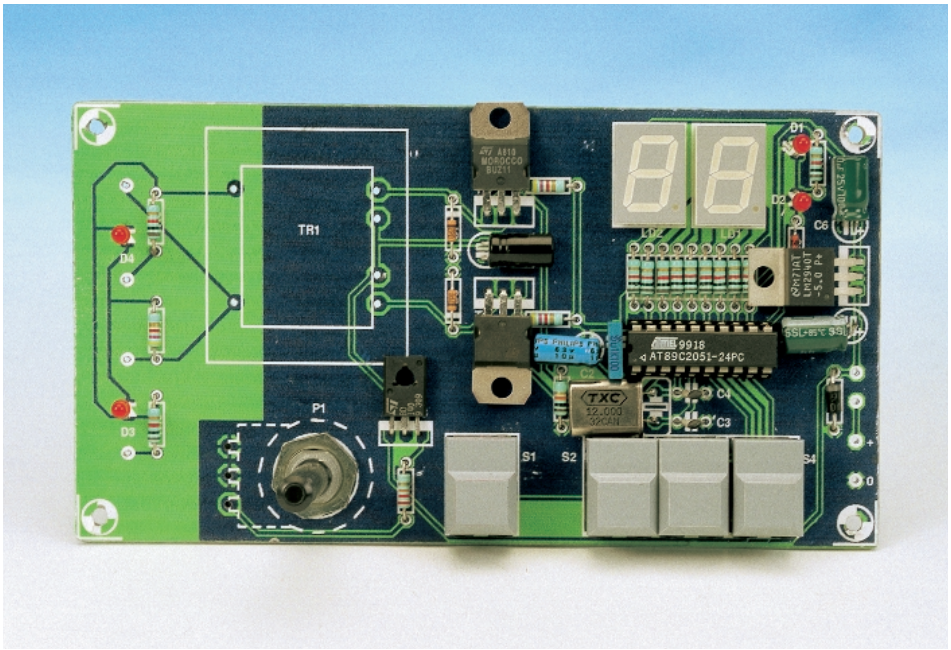


Figure 5. The transistors and the voltage regulator are mounted flat on the board, in order to limit the height of the assembly.

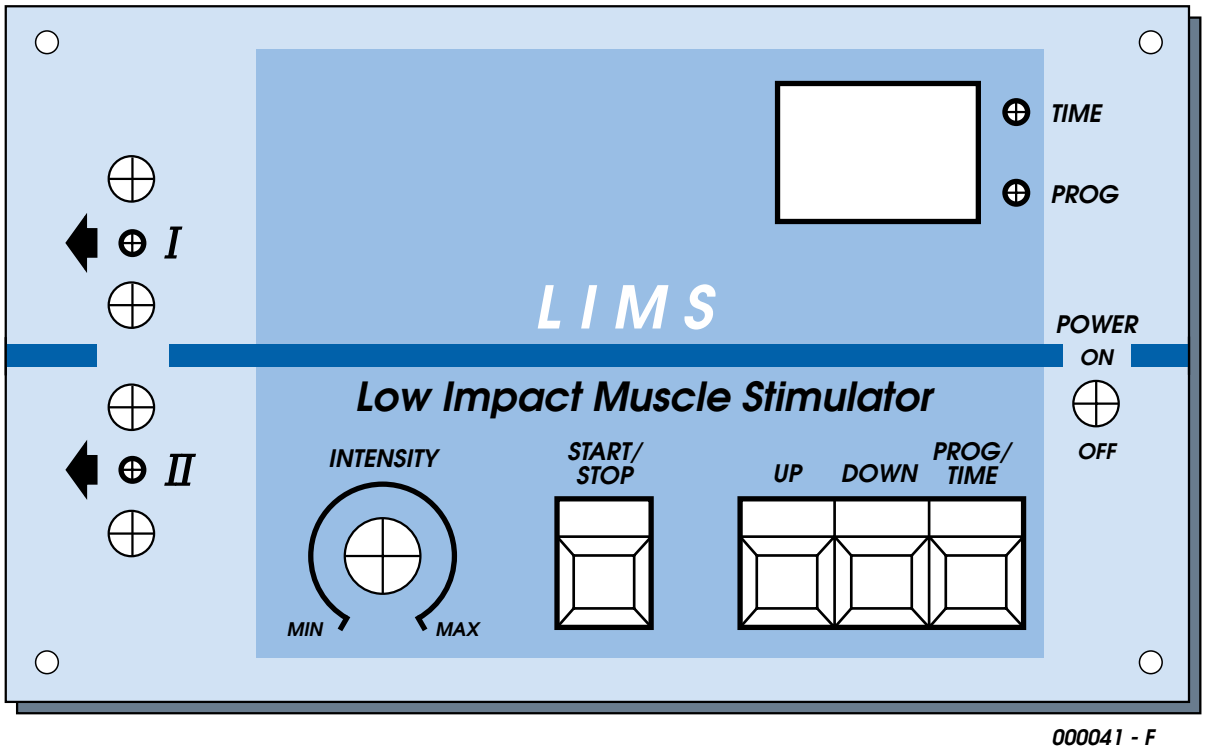


Figure 6. This front panel design matches the TEKO enclosure noted in the components list.

the electrode surface and the skin (don't use distilled water, since it doesn't conduct electricity). The separation of the two electrodes must be determined experimentally; it will usually lie somewhere between 5 and 30 cm.

If you plan to use the stimulator regularly, it is better to use a more durable material for the electrodes. You might consider purchasing 'real' electrodes with Velcro fastenings. These can usually be obtained in specialist medical equipment shops.

The connection between the electrodes and the stimulator can best be made using lengths of flexible wire (test lead cable) with miniature banana plugs at each end.

Initial operation

To check out the stimulator once you have built it, proceed as follows:

– Connect a 9-V battery, turn P1 completely to the left and switch on the device with S5. The 'program' LED D2 must now be on, and 'Lims' (for 'Low-impact muscle stimulator') should appear on the display, followed by a static dash. The dash means that no program has been selected. If this display indication appears, you can be practically certain that the processor part of the circuit is working properly.

– Press on the 'up' button. A '1' should appear on the display. Each time that the 'up' button is pressed again, the program number will be increased by one, while pressing on the 'down' button will reduce it by one.

– Press the 'program/time' button S4. A '0' will appear on the display, and the 'time' LED will be lit. Choose a time between 1 and 19 minutes using the up and down buttons.

– Press on the 'start/stop' button. A line segment will start to run around on the display in the form of an '8', as a sign that the circuit is active.

– Turn the amplitude control P1 slowly to the right. LEDs D3 and D4 should light up at a certain point, and they should become brighter as P1 is turned further to the right.

– If everything is OK up to now, turn P1 back to its minimum position (left) and switch off the stimulator with S5.

– Connect two electrodes to the stimulator, and clamp them to your leg (for example) about 10 cm apart. Switch on the stimulator, and select program 8 and a time of 2 or 3 minutes.

– *Check that P1 is turned to the minimum level*, and then press the start/stop button. You will not feel anything at this point.

– Turn P1 slowly to the right. At a certain point, you should feel a slight tingling. Cautiously turn P1 further until the desired intensity is reached. Don't overdo it!

The circuit also has a test program for checking that it is working properly. This can be run by holding the up and down buttons pressed while you switch the unit on with S5. If you then press the start/stop button, all segments of the display should light up and LEDs D1 and D2 should blink. You can exit the test program by pressing on the program/time button, which causes the software number to appear on the display.

(000041-1)

Text (Dutch original): S. van Rooij

Universal Parallel Input/Output for PCs

Hardware design by D. Aggelos

Here's a simple project with Windows control software that allows you to control up to eight relays or other actuators, and read back an equal number of input lines, all by means of the parallel port on your PC.

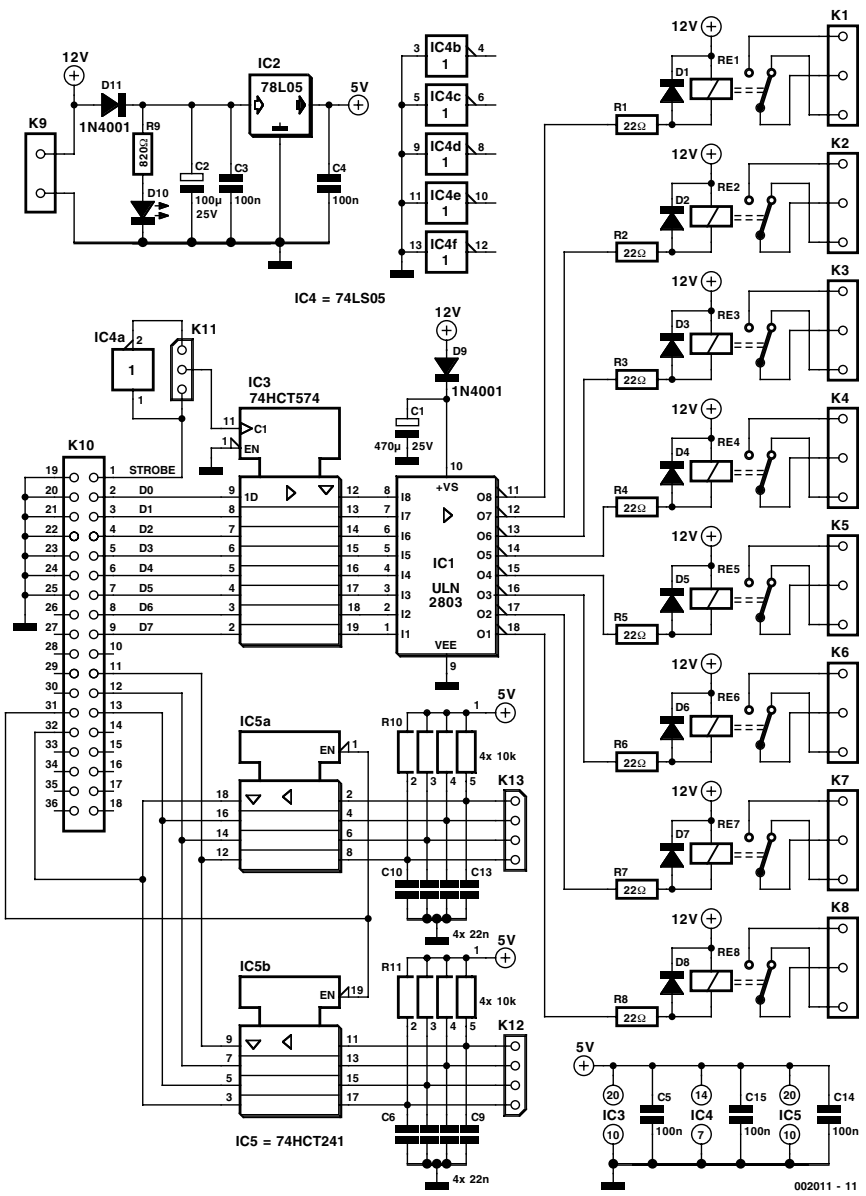


Figure 1. UPIO circuit diagram.

The UPIO software and hardware (a board attached to the PC's parallel port) allows you control up to eight relay outputs using a Windows-style user interface designed for user-friendliness. UPIO also allows you read back logic states on input lines.

Circuit description

The circuit that belongs with the UPIO program is very simple and consists of a few low-cost and easy to find parts. The circuit diagram is given in **Figure 1**. A tri-state buffer type 74HCT241(IC5) arranges all input contact reading. An 8-bit latch type 74HCT574 (IC3) is used to preserve the output state during instruction execution. The third essential component is a power driver type ULN2803 (IC1) which enables output relays Re1-Re8 to be actuated and de-actuated.

Because the LPT input port consists of just four bits (one nibble), two subsequent readings are required from the HCT241 to obtain the whole (8-bit wide) input word. Supply power comes from an external mains adaptor with an output of approximately 12 volts DC. An on-board regulator type 78L05 (IC2) provides the HCT and LS ICs with a 5-volt supply rail. Note that the ULN2803 and relays are powered from the unregulated 12-V supply. Obviously, the mains adaptor should be able to supply the necessary current (all relays may be actuated at the same time!).

The PCB

The PCB for this project was redesigned to reflect Elektor style and standards from artwork originally supplied by the author. The resulting design is shown in Figure 2. The ready-made board is available through our Readers Services as item 002011-1.

A PCB-mount 'blue ribbon' 36-way Centronics socket is used at the input to allow easy connection to your PC's printer port. At the 'output' side of the board, 3-way screw terminal blocks (connected directly to the relay contacts) allow the user to select N.C. (normally closed) or N.O. (normally open) type contacts. The digital inputs are divided over two 4-way SIL connectors, K12 and K13.

COMPONENTS LIST

Resistors:
R1-R8 = 22Ω
R9 = 820Ω
R10,R11 = SIL array 4 x 10kΩ

Capacitors:
C1 = 470µF 25V radial
C2 = 100µF 25V radial
C3,C4,C5,C14,C15 = 100nF
C6-C13 = 22nF

Semiconductors:
D1-D8 = 1N4148
D9,D11 = 1N4001
D10 = LED
IC1 = ULN2803
IC2 = 78L05
IC3 = 74HCT574
IC4 = 74LS05
IC5 = 74HCT241

Miscellaneous:
K1-K8 = 3 way PCB terminal block, raster 5mm
RE1-RE8 = PCB mount relay, 12V, e.g., Siemens V23040-A0002-B201
K9 = 2-way PCB terminal block, raster 5mm
K10 = 36-way Centronics connector, PCB mount
K11 = 3-way SIL pinheader with jumper
K12,K13 = 4-way SIL-header
PCB, order code **002011-1**, see Readers Services page and Elektor website.
Disk, contains all project software, order code **002011-11**, see Readers Services page and Elektor website (free download for subscribers)

Tailor-made process control

Originally, the author supplied rather simple control software. By means of a 'program-style file', created by the application, it was possible to produce a simple 'Output Flow' with user-defined timing. In addition, output states could be checked by reading up to eight contact-type inputs. Together with the output control this allows 'Hold' or 'One-Shot' action to be implemented, both are well-established features in flow control systems.

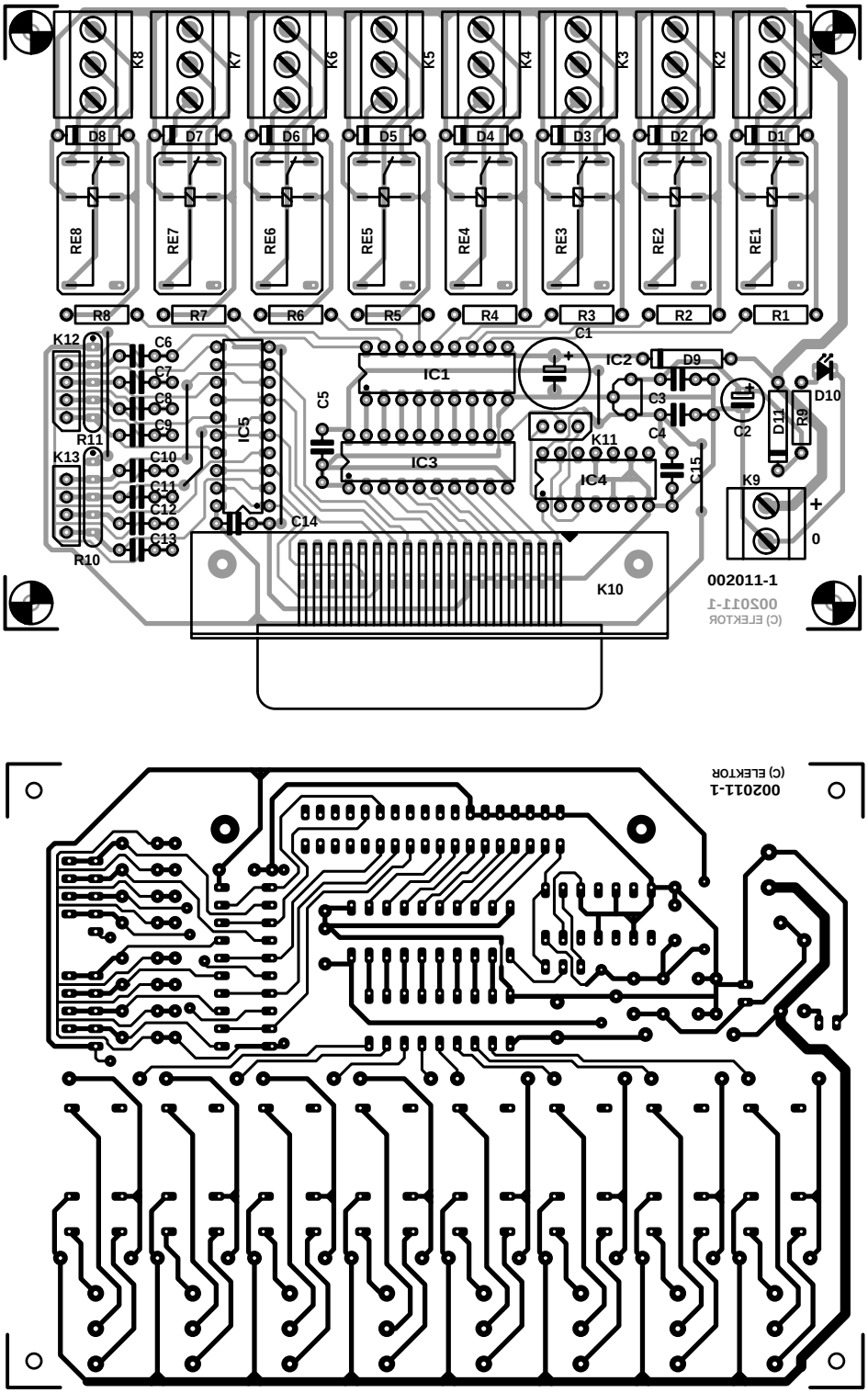


Figure 2. PCB design for the UPIO hardware (board available ready-made).

Talking to ports under Windows 95/98/NT

Having played around with the original software for a while it was felt that this was rather unrefined and unreliable. So our staff designer Luc Lemmens set out to write an improved control program for the UPIO hardware. Let's look at the underlying principles.

Under DOS and Windows 3.1, it used to be relatively easy to control PC hardware directly using higher programming languages like BASIC or Pascal. All that was needed at that time was the odd IN/OUT instruction in BASIC, or its PORT equivalent in Pascal.

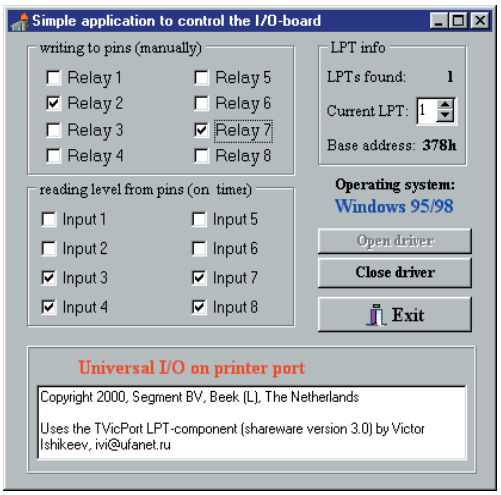


Figure 3. Screendump showing the UPIO control software in action.

In later versions of the Windows GUI, (release 95 and later), the user is increasingly 'isolated' from the hardware, and it is no longer possible (and even forbidden under Windows 95) to communicate directly with various registers and memory locations. This kind of 'Smalltalk' is completely left to the operating system in conjunction with various device drivers. Writing your own device drivers is arduous to say the least, so it is often better to go on the Internet and see if someone else discovered the wheel for you. In this case we landed at www.entechtaiwan.com where we found a (shareware) LPT component called TVicPort written by Victor Ishikeev. This component is available in versions for Delphi (version 2 through 5), Java Builder (version 1, 3 and 4), Visual BASIC (version 6), Borland C++ and MS Visual C++ in Windows 95/98 and Windows NT.

As could be expected in this day and age of object-oriented programming, the printer port is no more than an *object* with all sorts of *properties* and responding to all sorts of *events*. Using this concept the programmer has easy access to the object (in this case the printer port), while control is exerted via *methods* associated with the relevant component. A brief discussion of the methods and properties used in the example program (a Delphi 2 application) will be given below.

The really essential procedures are *OpenDriver* and *CloseDriver* which, you guessed it, open and close the TVicport respectively. The logic variable (property) *ActiveHW* flags if the port is open or closed, or, in actual fact, if the driver is available or not.

The property *LPTNumPorts* indicates the number of printer ports in the system, while *LPTNumber* determines which LPT is being controlled by the driver. Finally, *LPTBaseAddress* shows the base address of that particular port. The component also has a property called *Pin*, an array of 25 logic variables (bits) representing the logic levels (states) found on hardware port pins. By means of this array programmers have easy access to the pins on the printer port. You only have to be cautious which pin is input or output only, or bi-directional.

HWTest and the example program

The HWTest program as supplied on the project disk illustrates the practical use of this component in many ways. In fact, we'd say it is a perfect starting point for your own applications.

Naturally the example program (also supplied on disk) was derived from HWTest. The 8 relays on the UPIO board may be switched on and off by ticking the relevant boxes. The ticks in the panel below the relay boxes indicate if a digital input is high or low. If the PC contains several printer ports, the thumbwheel switch allows a different port to be selected. As a matter of course, the printer port control will only work if the driver has been successfully loaded by pressing the 'Open Driver' button. A screendump of the UPIO control program is shown in Figure 3.

Tools used for the project

For the original UPIO.exe application, the author used Borland Delphi development environment (Version 1-16 bits), which generates the '.dpr' application program files, the '.pas' unit files, the '.opt' and '.res' files. All original files for the UPIO project, as well as the refurbished version of the control software may be found on a floppy disk with order code 002011-11. It should be noted that the source code file as we received it from the author is supplied 'for information only' — it can not be edited because the LPT component is missing. Also, because of various problems we do not recommend using the original '.exe' file.

As a matter of interest only, the circuit diagram was originally designed using 'Protel' schematic capture, while 'Advanced Schematics' software was used for the PCB design. Both the circuit diagram and the PCB artwork were redesigned by Elektor Electronics for the purpose of this article.

(002011-1)

Design and software editing: L. Lemmens
Text editing: J. Buiting

Robots in the automobile industry

multifunctional manipulators for tedious work

By Bernhard Krieg

Most robots in the world are used in manufacturing operations: assembly and inspection, processing, and materials handling. Their use in assembly and inspection is accelerating in line with the increasing cost of manual labour. Robots are the culmination of the automated production line. The first industrial robot was installed (and its programmable manipulator patented) about 40 years ago in the USA. Basically, a robot is controlled by a computer and carries out the received instructions by teleoperators (mechanical manipulators).

For as long as man has had to work, he has sought to unload some of the burden of labour to mechanical devices. It was not until the Industrial Revolution, however, that large-scale mechanization could take place. Once division of labour, that is the restriction of the activity of each labourer to one specific, repetitive task, had been adopted, it was a relatively simple step to develop machines (first steam, and later electric) to carry out these tasks.

Another step toward automation was taken with the introduction of assembly lines, which used a conveyor belt to move a job in stages from one worker to another.

Automatic transfer systems, developed during the Second World War, combined assembly lines with mechanization. However, true automation could not come about until the development of feedback systems. Such systems enable machines to monitor their own output, compare it with a set of standards, and adjust their operation accordingly if and when necessary.

A robot is such a machine to replace human effort. Some idea as to the effectiveness of robots may be gathered from the fact that in a modern car factory, such as the Daimler-Chrysler plant in Rastatt, Germany, 5000 workers and 380 industrial robots produce some 200,000 private cars per year.

Twenty-two hours

In the Daimler-Chrysler plant, it takes 22 hours for a complete car to roll off the production line. What part of the work is carried out by robots?

Clearly, before final assembly can be started, materials and other parts have to be moved from one location to another. Most of this involves robots picking up these parts from one conveyor and placing them on another.

In the assembly hall, robots fasten the panels to the chassis – up to the doors, but no interior work is done as yet. There are 290 different body parts and these are spot-welded together at 4000 different locations (**Figure 1**). The assembly hall is spacious enough to allow the 'herringbone' type of production line. In this, small subsidiary transfer lines (conveyor belts with ancillary production stations) are positioned at an angle to the main belt. This method of working ensures that the car body shell moves along the main production line at a fairly constant speed.

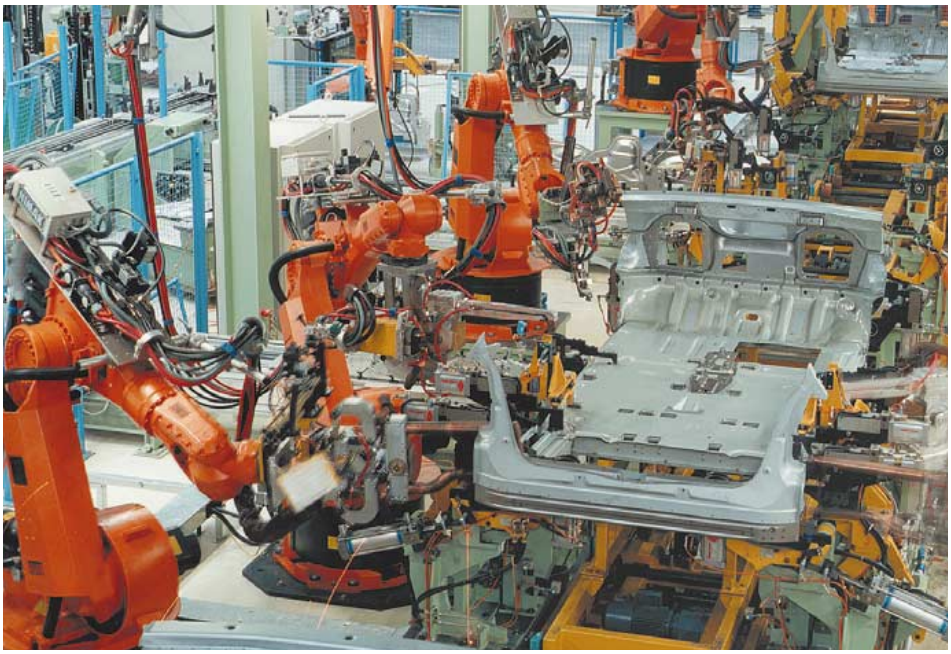


Figure 1. In the production hall 330 robots manufacture the chassis from 290 different parts. Shown is the completion of the understructure of the body.

Surface finishing

Robots are also used to paint the completed shell: naturally under clean-room conditions. Because of environmental considerations, only water-based lacquers – without a nickel or nitrate pre-treatment – are used. The lacquers applied by a dip and rotate process contain no lead. The final three films of paint are applied electrostatically to reduce overspray. Drying between the applications of the various layers is not necessary.

Since the first (body-colour) film also provides protection against stones thrown up from the road and colour change through ultra-violet light, the usual plastic (protective) film is

Figure 2. Application of the final (clear) solvent-free lacquer by a paint robot.

Figure 3. Owing to customers' requirements, completion of the interior of the car requires more workers than robots, that is, this stage is more labour intensive than the remainder of the car.

not needed. The final layer is a clear lacquer that is free of polluting solvents (**Figure 2**).

Once the robots have carried out their allotted tasks, the shell goes to the 'white room'. In this, human operators inspect the paint quality in a bright, completely white space. In this environment, no blemish escapes the inspectors – no robot is perfect!

Assembly

During assembly, the doors are attached to the shell, after which the car is completed to customers' requirements. The work is carried out by robots as well as people – in fact, it is the most labour-intensive stage of car manufacturing (**Figure 3**).

The assembly line contains a mixture of different models: right- or left-hand drive; automatic, semi-automatic or manual transmission; different sizes and types (petrol/diesel) of engine; and interior finishes. All more or less heavy labour is carried out by robots. For example, they fit the windscreen (US: windshield) with millimetre precision, build in the seats, and attach the bootlid (US: trunk lid).

Whereas parts are delivered 'just in time', the sub-systems are delivered 'just in sequence'. Fifty per cent of the parts required for assembly are delivered directly to the main assembly line. Five hundred or so workers fit the bulky, subject-to-model items, such as seats, door fittings, and cable harnesses. These items are fed on to the belt by a conveyor trolley.

Finally

And so, after 22 working hours, a new customized vehicle rolls off the assembly line. A box containing the data of all customer requirements is attached to the chassis so that these requirements can be read again during a final inspection and test.

[000035]

Excursus

Robots

The Robotic Industries Association defines an industrial robot as 'a reprogrammable, multifunctional manipulator designed to move materials, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks'.

The name robot is taken from the Czech 'robota', meaning forced labour. Modern use of the word stems from Karel Capek's 1921 play Rossum's Universal Robots, in which mass-produced robots in human form turn on their creators and exterminate them.

Robots used by Daimler-Chrysler

The robots used in the German Daimler-Chrysler plant mentioned earlier are not typical of those used generally in the automobile industry, but they serve well as an example. They are KUKA Types IR360/125 and IR360/150. The final three digits indicate their maximum loading in kilograms. These robots do not need anti-torque weights since hydraulic mass equalization is provided. This facility also enables the robot to correct its movements if these deviate from the programmed patterns. Since these robots are accurately 'cloned', there is no problem in substituting one by another or in replacing parts.

Manufacturing

The first requisite in the manufacturing of cars is an accurately controlled flow of materials into the assembly plants. Many car manufacturers produce most of the required components and materials themselves. However, a number of components, for example, tyres, batteries and dashboard instruments are generally procured from outside sources.

Worldwide, most car manufacturers use the same kind of assembly process (although the Daimler-Chrysler plant mentioned earlier in the article uses a method whereby the body and frame are assembled as a unit).

Normally, there are two main assembly lines: body and chassis, on both of which robots are used extensively. On the first, the body panels are welded together, the doors and windows installed, and the body painted and trimmed (upholstery, interior hardware, wiring). On the second, springs, wheels, steering gear, power train (engine, transmission, drive shaft, differential), brakes and exhaust system are installed.

The two lines merge at the point at which the car is finished but for minor items and inspection and testing (by human operators).

Over the years, assembly lines have been refined by automatic control systems. In Britain, automatic transfer machines were introduced as early as 1950. The first large-scale automated installation was taken into use by the Ford Motor Company engine plant in Detroit, USA, in 1951.

Industrial robots

Virtually all industrial robots provide a substitute for human labour that is repetitive, hazardous or uncomfortable for the worker (spray painting, spot welding, arc welding, operations involving lasers), or where the task requires the handling of a part or tool that is heavy or awkward or both to handle. Moreover, the same robot can be used on consecutive shifts throughout a 24-hour period, seven days a week, 365 days a year, interrupted only for maintenance. To carry out these tasks, robots must be

- freely programmable
- servo controlled
- designed to have at least three axes
- equipped with grippers and tools
- designed for maintenance and processing tasks

In all cases, however, a robot is normally used only when this is economically sound. Since in the modern world, labour is getting more and more expensive (at roughly 5 per cent per year), and robots are getting less and less expensive (by roughly 5–8 per cent per year), more and more robots will be used in the world's manufacturing plants.

Types of teleoperator

Basically, there are several types of teleoperator (mechanical manipulator) used in robots, but all are made up of a series of links and joints. There are five basic types of joint to construct the manipulator: two are translational and three are rotational.
The teleoperator is usually divided into two parts: arm-and-body, and wrist. The various possible axes (degrees of freedom of movement) of a 6-joint manipulator are shown in **Figure 1**. Axes 1, 2, and 3, are the principal joints of the teleoperator that enable turning, stretching and bending, actions comparable to the arm sweep, shoulder swivel, and elbow movement of the human arm. Axes 4, 5, and 6, enable movements akin to those of the human wrist: yawing, pitching, and rolling.

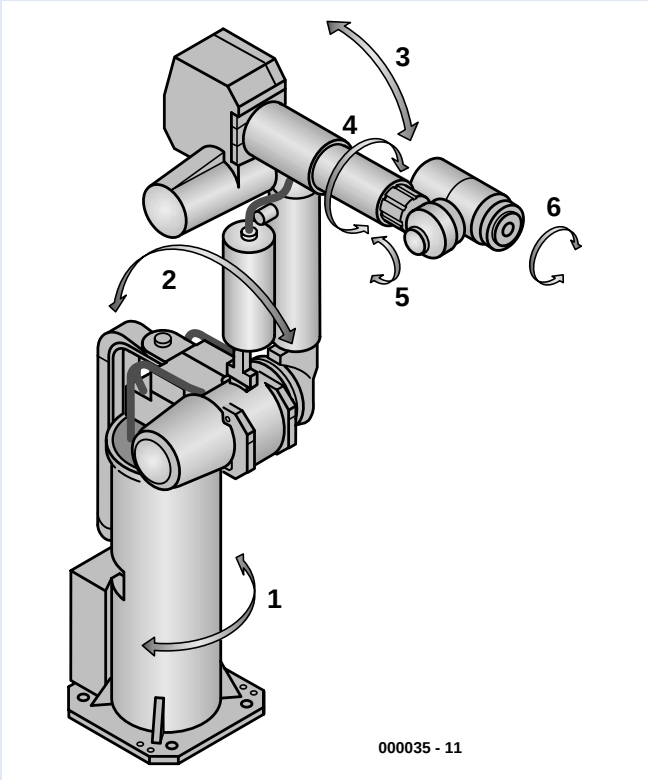


Figure 1. A robot with 6 axes (degrees of freedom).

The principal axes position the wrist joints in space. These joints can therefore orientate grippers or tools in space in such a way that processing or maintenance tasks can be fulfilled. When, for instance, a screw is being turned, joints 1–5 are used to bring the screwdriver, that is, joint 6, into the correct position. Compared with this manipulator, those with three or four joints are clearly less versatile, although they may well be perfectly all right for use with less demanding tasks.

Linear manipulators

Linear manipulators, also called Cartesian manipulators, usually have three principal joints (Figure 2). These enable up-and-down, side-to-side, and crossways movements. Often, there is a fourth joint that enables turning a tool around the z-axis.

Figure 2. Linear teleoperator with three translational joints.

The work space, also called work envelope, of a linear teleoperator is cuboid. Modular adaptation of the possible travel of a single joint enables the envelope to be modified. Axis 1 (x-axis) enables longitudinal travel of up to more than 15 metres (50 ft). Owing to the maximum permissible mechanical load the travel of axes 2 and 3 (crossways and up-and-down respectively) is restricted to 2–3 metres (7–10 ft).

Roll manipulators

Roll or cylindrical manipulators normally have four degrees of freedom. These are provided by three rotational (or spin) joints that can revolve around the z-axis and one translational joint that operates along the z-axis. Often, however, only joints 1 and 2 are rotational and joint 3 translational. The fourth joint to turn the tool along the z-axis is then integrated into, or fitted to, joint 3. The work envelope of a roll teleoperator is cylindrical, or very nearly so.

Yaw manipulators

Yaw teleoperators are derived from swivel manipulators: three wrist joints are added to the three main joints (Figure 3). The added joints are arranged so that they can be yawed (that is, moved in the horizontal plane, whence the name of the manipulator).

The work envelope of this type of manipulator is cylindrical. Because of the differential arrangement of joints 1–3, the cross-

Figure 3. Yaw manipulator.

section of the cylinder may be circular or cardioid. The diameter of the work envelope depends on the length of the links connecting the rotational joints: currently, diameters of up to 5 m (16 ft) are possible.

Pitch manipulators

Pitch teleoperators are widely used in industrial applications (Figure 4). Three wrist joints are added to the rotational joints (one for vertical and two for horizontal movements). The wrist joints pitch (that is, move in a vertical plane, whence the name of this type of manipulator).

The work envelope of this type of manipulator is basically spherical. The extent of the envelope, particularly the inner space of the hollow sphere, is strongly dependent on the degree of freedom of the various joints. In the case of large robots, the diameter of the sphere may be as much as 5.5 m (18 ft).

Figure 4. Pitch manipulator.

Special manipulators

Apart from the basic types of teleoperator discussed, there are many types of special manipulator, depending entirely on the application. Some provide improved access, others more rigidity, and yet others more freedom of movement.

Operation and programming

Some industrial robots are operated and programmed with the aid of a special control panel (Figure 5). This control panel allows all necessary instructions to be input and the operation of the manipulators to be actuated. The keypad of the panel is divided into clear functional sections separated by coloured lines. Integrated light-emitting diodes (LEDs) guide the operator to the many input facilities. Pull-down menus and groups of user and application specific softkeys enable fast programming in a straightforward manner.

Another programming tool is a 6-D mouse. After this has been calibrated in accordance with the manipulator's co-ordinates, it enables the teleoperator to be moved in all six directions of freedom to the desired position.

Figure 5. Control unit and operating panel in enclosure.

To prevent accidental operation of industrial robots (which are powerful and fast), the actual movement of the manipulators is enabled only after the 'safe' (or 'consent') key on the control panel has been pressed. The switch operated by this key controls security functions after the robot has been set up. When the relevant switch section is open, the manipulator can move at speeds of more than 25 cm/s only when the 'safe' key is held depressed. When the operator releases the key or presses it to the end position (out of fear?), the robot control is switched off. During direct programming, the robot cannot be active.

Normally, however, industrial robots are programmed by special computers. The software in these computers can, of course, easily be replaced or modified for the manufacture of different products. The programming language is usually APT (Automatically Programmed Tools), but there is increasing use of Flexible Manufacturing Systems (FMS), which coordinate design, production control, and manufacturing in a site-wide computer network linked with LANs, such as Ethernet, MAP and Fieldbus.

Computer programming is followed on a VDU, which requires good spatial perception. Even when all co-ordination points have seemingly been programmed correctly, some may have to be

trimmed or corrected in a teach-in operation. Even programming with CAD/CAM software is invariably followed with such an operation.

Hardware

The numerical control unit of the robot consists of several electronic cards linked by a bus system. The central unit is the main processor that determines the overall co-ordinates of the robot control. Superimposed on this unit are joint-regulating processors that carry out the positioning and monitoring of the various joints. The performance of the main processor in current robot control units is comparable to that of, say, a Motorola Type 68040 or Intel 486.

The matching unit provides communication between the robot and peripheral equipment, for instance:

- interface to robot (reading and writing the data as to position of manipulators; driving the servo control; operating the brakes);
- binary interface to peripheral elements (control of grippers and conveyors; reading/writing of sensor data);
- analogue interface to processes (control of processes; reading/writing of analogue sensor data);
- interface to auxiliary processors (off-line programming system; master processor; PPS processor; CAD processor).

Communication often takes place over a 3-wire CAN (Controller Area Network) bus, because of the high transfer security and minimum cable requirements of this bus. (See Elektor Electronics Sept–Dec 1999).

The power unit provides supply voltages for the various electrical and electronic units.

Servos control the joints of the manipulator(s) in line with instructions from the robot control. Their use ensures that the movement of the teleoperators is arranged with great precision. The requisite three-phase motors are driven in such a manner that, depending on the amplitude of the current and the current/voltage phase relationship, the rotary moment is in precise accordance with the instructions from the robot control unit.

The protection unit is an important unit not only for the robot system, but also for the operator. It provides traditional voltage and temperature monitoring, an emergency stop switch for the hardware and software, and input diagnosis of all switch actions. The unit also monitors switching systems like the ‘safe’ switch and emergency stop switch, and the speeds during setting-up operations.

BASIC Stamp programming course (8)

Part 8 (final): the photophobic robot

By Dennis Clark

Phase 3: the photophobic robot and subsumptive programming

It would be interesting if the robot had some higher purpose than just wandering randomly around the room. Let's make a cricket-like behaviour, a bug that looks for a dark corner in which to hide. We describe the behaviour that makes our robot seek darkness and avoid light as *photophobic* which means 'fear of light'. Fear is a living being's emotion, but we can make our robot *appear* to fear light with this behaviour! Build the light sensor circuits in earlier instalments to use this code. To read the photocells we will use the Stamp II instruction *rctime*. Reading a photocell takes time, the darker it is, the longer it takes to read the cell. Because it takes so long to read a single photocell, we will want to break up the readings into different states, doing the math to determine whether or not the left or the right photocell sees brighter light also takes time. We don't want our robot to stop and think every time it looks at the light readings to the left and right, so again, these will be separate states. When we have made a decision, we want to turn in the direction of the darker reading. We should turn for a while, so we will be setting a duration for the turn as well as a direction. If we don't have a certain duration, our robot could easily jerk back-and-forth, which doesn't look like its being very decisive! This will be a more complex state machine than our previous ones because it has five states instead of two. Here is a list of actions that will need to be done:

State 0
  Read light level on left photocell
  Set state = 1
State 1
  Read light level on right photocell
  Modify this value by 1.5 because this cell reads a little lower than the other one
  Set state = 2

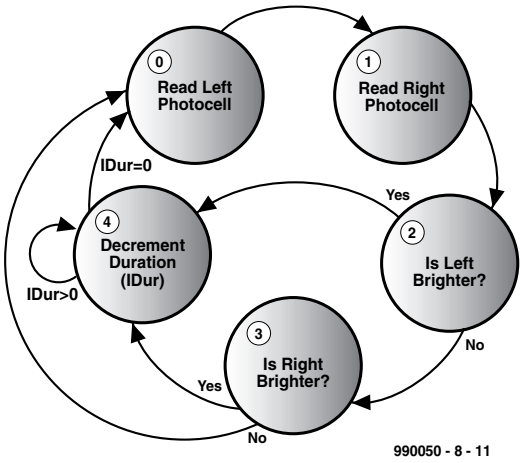


Figure 25. Proposed state machine for photophobic behaviour.

State 2
  Add margin to left reading
  If left is brighter than the right
    Set lDir = turn right (tr)
    Set lDur = 30
    Set lstate = 4 (next state is decrement)
  Else
    Set lstate = 3
State 3
  Add margin to right reading
  If right is brighter than the left
    Set lDir = turn left (tl)
    Set lDur = 30
    Set lstate = 4
  Else
    Set lstate = 0 (do nothing, both sides are about the same)
State 4
  Decrement lDur, lDur = lDur – 1
  If lDur = 0
    Set lstate = 0 (start over from beginning, we are done)
  Else
    Do nothing, come back to state 4 again for a decrement, we are still turning

Figure 25 shows the state machine we will be using for our *photophobic* behaviour.

This state machine was drawn only after all of the actions that were required had been written down and organized in a logical manner. Because we are changing variables and passing data between these states, we really should include that in the transition function arguments (the captions on the arrows). Sometimes it is easier to create the state diagram

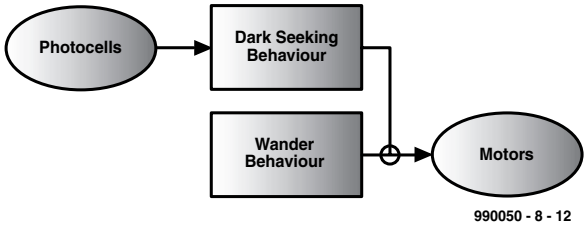


Figure 26. Wander/seek actions defined in terms of subsumptive programming.

and build the functionality of each state from that graph. There is more than one way to build a behaviour, this is just one suggested design.

Notice that the description of the states is pretty detailed. Your descriptions should be this detailed so that you don't forget anything that could be important. In State 1 a 'fudge' has been added to the reading. This is because no two photocells are the same, when these were measured, one read a little lower than the other, this modification evened out the photocells for comparison. This is something for you to check with your photocells too. Of course, since our state machine is quite a bit more complex, you can expect the code to be more complex as well. It is, but if you write the steps that need to be taken very carefully, in a programming-like language (as seen above), you can practically write the software straight from your state description. This 'almost' programming language is called *pseudo code*, it helps you to organize your thoughts logically. If you can't write it down, you can't program it! The actual Stamp II code for this behaviour is shown in Listing 13.

There are a few subtle short cuts in this code. The `tmp = pright >> 1` statement divides pright by two and stores the result in `tmp`, then the next line adds `tmp` to `pright`. This multiplies `pright` by 1.5 to compensate for its lower readings than the left photocell. The branch statement is a way to branch to a different section of code based on an index value, in our case that index value is our state.

Add the variable and constant column on the left to the top of your program. Add the *lightlook* behaviour subroutine to the end of your program. Add this behaviour to the *main* programming loop like this:

```
'set up for running
wstate =0    'initial wander
state
lstate =0    'initial photo-
phobic state

main:
    gosub wander
    gosub lightlook
    gosub act
goto main
```

What do you think will happen? The *wander* behaviour will set the variable *drive* to some random direction, however, the *lightlook* behaviour, knowing nothing at all about *wander* will then set *drive* to some other direction perhaps, if it sees a darker corner to run to. Finally, *act* will use the *drive* value to determine which motors to drive in which direction. So, the *lightlook* behaviour will have a higher priority than the *wander* behaviour because it gets the opportunity to set the *drive* variable after *wander* does. Are you starting to see the exciting possibilities with this type of robotic programming? Let's look at the subsumption network diagram in Figure 26 to see the behaviours we have just given our BoE-Bot.

Here is a challenge for you! Turn our little photophobic robot into a cricket that hides in the dark and chirps. You can start with this list of requirements:

- Both photosensors must read some low level of light (high reading = low light)
- When the low light threshold has been reached, the robot must stop there
- Use the last readings taken from *lightlook*, this means you won't need to take new readings
- The robot will only chirp when it is standing still in the dark
- This will be higher priority than the *lightlook* or *wander* behaviours
- If either sensor's value goes above the threshold, then the robot will move again

By using the last read light values that were taken by the *lightlook* behaviour we are violating the strict rule of independent modules. However, in the interest of saving the time that these readings would take, this is a small infraction. Besides which, this isn't a traffic law, and very little side-affect occurs from this minor violation. Feel free to add the *rctime* functionality if you wish.

Now your task will be to create a detailed list of actions that must occur in each state of this finite state machine. Then draw a state diagram, decide where this behaviour belongs in your robot's subsumption network and finally write the behaviour code and insert the *gosub* for the behaviour into the *main* loop. Make sure you initialize your state machine in the *Set up* section before the main behaviour loop!

Phase 4: the Avoid behaviour

We have random movement and photophobic behaviours programmed into our robot now. However, BoE-Bot does not have any way to avoid running into things. A behaviour that will avoid walls and furniture will certainly extend the usefulness of our little robot! Build the IR proximity detectors (IRPD) from an earlier instalment. Let's think about what we need to do to avoid collisions with objects. An enhancement to this behaviour would be to have the robot turn 180 degrees when it sees an obstruction directly in front of it. How would you go about creating this behaviour?

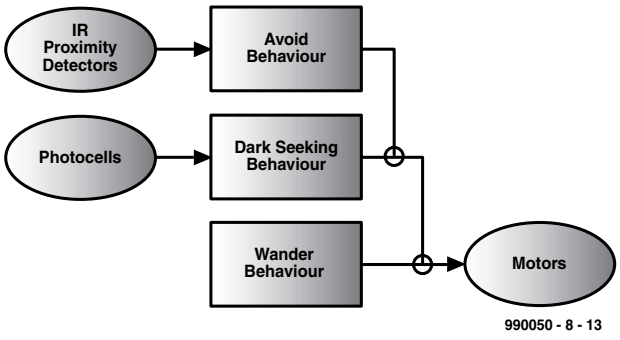


Figure 27. Subsumption network diagram for a photophobic robot.

```
Get IRPD readings
If first reading then
  Save it
Else
  If this reading = last reading then
```

Listing I3

```
'light looker vars and constants
LLIGHT   con     11       'left sensor
RLIGHT   con     4        'right sensor
pleft    var     word     'left value
pright   var     word     'right value
lstate   var     byte     'FSM state
lDur     var     byte     'how long to go
lDir     var     word     'where to go
LMARG    con    15       'light margin
lightlook:
  low LLIGHT              'set up for sensors
  low RLIGHT              'branch takes 200us
  branch lstate,[lread1,lread2,lcomp1,lcomp2]
    lDur = lDur - 1       'state 4 decr duration
    drive = lDir 'correct direction
    if lDur > 0 then lDone1       'still counting
        lstate = 0                'restart FSM
lDone1:                           'done
    return
lread1:                           'state 0
    rctime LLIGHT,0,pleft         'get left value
    lstate = 1                    'go next state
    return

lread2:                           'state 1
    rctime RLIGHT,0,pright        'get right value
    tmp = pright >> 1             'compensation
    pright = pright + tmp
    lstate = 2                    'go next state
    return
lcomp1:                           'state 2
    tmp = pleft +LMARG            'set threshhold
    if tmp > pright then lDone2   'left not past
                                  'threshhold
        lDir = tr                 'bright to left
        lDur = 30                 'for a while
        lstate = 4                'go decr state
        return
lDone2:
    lstate = 3                    '2nd compare
    return
lcomp2:                           'state 3 compare
    tmp = pright +LMARG  'set threshhold
    if tmp > pleft then lDone3'not past
        lDir = tl                 'bright to right
        lDur = 30                 'for a while
        lstate = 4                'go decr state
        return
lDone3:
    lstate = 0                    'none past
    return
```

Choose direction and set *drive*
Clear reading history
Else
Save this reading

This one isn't as complex as the *lightlook* behaviour; going around something is a simpler behaviour than actively seeking something (dark) is. Some things are so simple that no state machine really needs to be built. One could say that this behaviour really does have two states, the first just takes an IRPD reading, the second

Listing I4

```
'IRPD vars and constants
ileft    var     in9      'IR LED outputs
iright   var     in0      'i=(see code)
IEN      con     5        'enable for 555
ilast    var     byte     'hit counter
avoid:'IRPD routine
  High IEN                'enable 555
  i=0
  i = ileft * 2 + iright  'read IRPD
  low IEN                 'disable 555
  if ilast = I then ickit 'two reads agree
    goto iDone            'just first read
ickit:  'This line chooses new direction
  lookup i,[rr,tr,tl,drive],tmp
  drive = tmp
  i=0                     'clear history
iDone:
  ilast = i               'new history
return
```

takes another and sees if they agree. However, these two actions are so closely coupled that it would be difficult, and unnecessary to separate them out.

Our code for *avoid* is shown in Listing 14.

Math in the Stamp II is evaluated from left to right, so you either must be careful of your order of operations or use explicit parentheses to force the correct order. The IR demodulators that we are using detect a signal by sending a 'low' or '0' back on that I/O port line. Therefore, with our code above, a 3 means no detection, a 2 means detection on the right, a 1 is a detection on the left and a 0 is detection on both lines. Our *lookup* instruction includes the old *drive* value in its list, this is done so that we can change nothing if there is no detection, and not modify the *drive* value at all. The *avoid* module will continue to change our robot's direction until there is no obstacle detected. This will look like a smooth search until the path is clear!

To add this behaviour, place the variables and constants block near the top of your current program and put the *avoid* subroutine in your behaviours code section. Since we want *avoid* to have higher priority than any other behaviour we have done so far, place it in the *main* loop as shown below:

```
'set up for running
wstate =0  'initial wander state
lstate  =0   'initial  photophobic
state
ilast =0   'initial avoid history
value

main:
    gosub wander
    gosub lightlook
    gosub avoid
    gosub act
goto main
```

Figure 27 shows our subsumption network diagram with all of our behaviours included.

Subsumption networks are quite useful to explain the potential activity of our robots to others. They can be used to predict behaviour or to compare behaviours of other robots with our own. Reading these diagrams can be easier than studying other peoples' programs! This type of programming can be used in any application that needs good response to external stimuli that is not dependent on precise time intervals.

Here is another challenge for you and your BoE-Bot projects. Add a behaviour that will cause your robot to backup and turnaround when it bumps into an object. This will require you to build a bumper and connect it to an I/O port on the Stamp II first. Then you will need to decide on a list of actions that need performed, break these actions into states and decide what you will need to remember. You will also need to decide what priority a bumper reaction should have in your subsumption network. You now have a set of tools for programming these behaviours into your BoE-Bot, go have fun!

(990050-8)

Internet

http://www.parallaxinc.com — BASIC Stamp Manual Version 1.9, BASIC Stamp DOS and Windows Editor, program examples. International distribution sources.

http://www.stampsinclass.com — BoE documentation, Robotics curriculum, BoE-Bot *.dxf and *.dwg drawing formats, discussion group for educational uses of BASIC Stamp.

chucks@turbonet.com — creator of the BoE-Bot and co-author of this series. Technical assistance.

kgracey@parallaxinc.com — co-author of this article. Technical assistance and questions about the educational program.

http://www.milinst.demon.co.uk — UK distributor of Parallax BASIC Stamp.

Robot parts

Order electronic and mechanical parts on-line

Designing and programming robots is one thing, and soldering and debugging another, but when it comes to actually making and assembling mechanical parts many of you will fall silent. Help is on the way, though, via the Internet.

Although most suppliers of robot kits and parts seem to be based in the USA, that need not be a problem since e-commerce comes to our rescue and credit card payment is now available in most countries.

Mondo-Tronics Robot Store is a large on-line shop offering many Lego Mindstorms components. Here we also found information on robot clubs, robot rallies and photographs of less than ordinary robots. Have a look at

http://www.robotstore.com/

Acroname is a US company aiming at helping customers build robots using electrical and mechanical parts and many other ancillaries. Although only a modest number of parts are available, we came across some really interesting items!

http://www.acroname.com/

Another site you can not afford to miss is

http://www.hobbyrobot.com

where books on robots, microcontrollers and software may be ordered. This site we found particularly well laid out, providing clear pictures of all sorts of robot parts.

Robot Shop is another name that leaves nothing to be desired as far as clarity is concerned. This on-line business also supplies a variety of robot parts. When we logged on, the site was being refurbished. None the less, our impression was favourable:

http://server17.hypermart.net/renelectronics/index.htm

Australia-based Robot-Oz supplies parts as well as kits. Their site at

http://www.robotoz.co.au/~robotoz

sure is worth visiting, if only for the nice pictures!

MrRobot supplies robot kits, mechanical parts and electronic components. Here we found, among others, Talrik robots from Mekatronix. In particular Talrik II, the round model looked interesting and is supported by lots of extensions. Visit the MrRobot shop at

http://www.mrrobot.com/

Other companies and web sites we should not forget to mention here include

RobotoKitsDirect

http://www.owirobot.com

Solarbotics

http://www.solarbotics.com

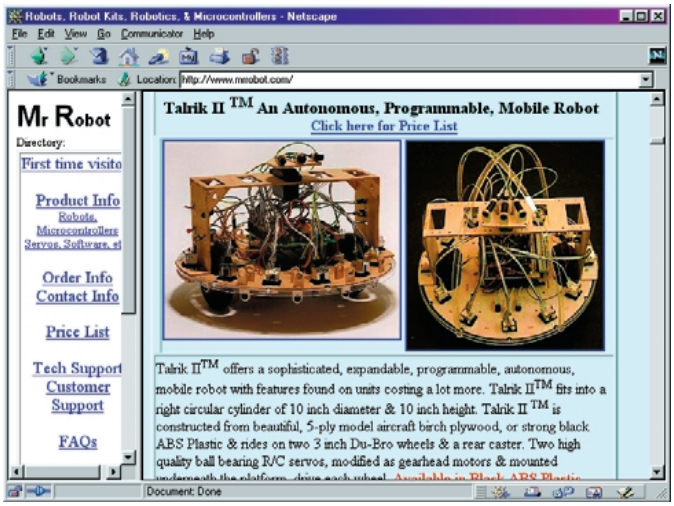
Parallax

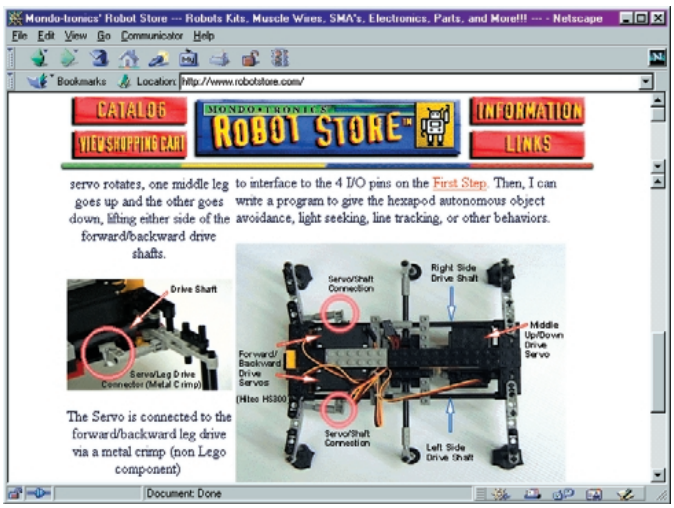
http://www.stampsinclass.com/products.htm

Lynxmotion

http://www.lynxmotion.com

005032-1





design robots with your PC

Lego Robotics Invention System

By H. Steeman

The Lego Robotics Invention System has been available for a while already. This system enables every computer hobbyist to design, build and above all program robots to his heart's content.

This kit contains more than seven hundred pieces, including an intelligent module (RCX) that can be programmed wirelessly from the PC. This is a wonderful robot system for any hobbyist or PC fan. And of course it's only fitting that we at *Elektor Electronics* should devote considerable attention to this development. Robots in general, and the Lego system in particular, will be featured in the coming months.

Long-term development

The development of the intelligent module that forms the heart of the robot development system was not an overnight affair. Lego devoted more than ten years in total to the realisation of the design of this module. After the system first appeared on the market in America around one year ago, a number of improvements were made. This made it possible to introduce a polished product into the European market in 1999. The design and capabilities of the intelligent module, which Lego calls the RSX 1.0, have received a lot of admiring and respectful attention. The NASA Discovery mission STS-96 even carried a robot

Figure 1. The RCX module is an innovative Lego block, packed full with electronics, that allows intelligent models to be built.

built using the Robot Invention System as a supplementary payload. This came about as part of a project to stimulate the interest of young people in information technology.

The developers of the RCX module worked together with employees of the Media Lab of the Massachusetts Institute of Technology in the course of the development project.

There is broad international admiration for the Lego concept. This system, which is based on ABS plastic building blocks produced by the Danish Lego company, was recently named the product of the century by the leading American journal *Fortune Magazine*. With the introduction of the Robot Invention System (RIS for short), which by the way is part of the much larger Lego MindStorms program, Lego has brought an entirely new dimension to the use of PCs. A PC owner can use these kits in a very adult and professional manner to build robots that can execute up to five programs and are able to react to stimuli from their surroundings. The programs can be written using a graphic programming environment specially developed by Lego, as we will see in subsequent articles. An advanced programmer can also program the RCX module using a professional programming environment such as Visual BASIC, Visual C++ or Delphi. The necessary library of basic routines has been published on the Internet.

Figure 2. An Internet portal has been set up especially for the MindStorms systems. Users can exchange experience and programs via this site.

System requirements

The PC to be used must meet a certain number of requirements. The operating system should be Windows 95 or 98, and the system must have an 8× CD-ROM drive, a Pentium processor (166 MHz), an SVGA video card (800 x 600 pixels and 1 MB of video memory) that supports DirectX, a 16-bit SoundBlaster-compatible sound card, a 9-pin serial port and at least 16 MB of working memory. There must be at least 70 MB of free space on the hard disc drive. Every standard PC model should have no trouble meeting these requirements. For optimum use of the RIS, access to the Internet is necessary.

The sensors currently available include contact sensors (miniature switches built into Lego blocks), a combined light sensor and reflective sensor, a rotation counter for measuring distances and a temperature sensor. It is thus possible to sense analogue ambient variables, such as the temperature of the surroundings and variations in light intensity, as well as digital values such as contact/no contact. The RCX module has an infrared communication port that can be used transfer information between the module and a PC. This port can also be used to allow several RCX blocks to communicate with each other.

Each RCX module has three pulse-width modulated outputs that can be used to actuate motors and lamps. An integrated buzzer can be used to generate an audible signal. There is also a built-in liquid-crystal display to provide information to the user. Figure 1 shows an RCX module that is connected to two motors and three sensors. One of the outputs of the RCX module shown in the photo is not used.

The Robotics Invention System combines the strength of Lego, which is providing components that can be used to build mechanical constructions, with the intelligence of microelectronics and the computational power of the PC. The Internet has also been brought into play, in order to provide users of the RIS with a common meeting point in the form of a dedicated 'portal'. This is located at the web address *http://www.legomindstorms.com*.

What's in the box?

The yellow RCX module is the most eye-catching part of the kit. This oversized Lego block still has some resemblance to the famous standard Lego block, and it also can be built into the models in the same manner as a conventional block. It contains a complete microcomputer system with its own operating system, which can be downloaded from a PC.

The heart of this system is a 64-pin Hitachi microcontroller from the H8 family, which works with a 16 MHz clock. The microcontroller has 16 kB of internal ROM and is supported by 32 kB of RAM.

Figure 3. The PC software shows the user everything that the system can do.

The RAM has a battery supply with a limited backup function. As long as the six penlight batteries (normal or rechargeable) supply power, the program is held in the memory. The content of the memory is retained for one minute without power while the batteries are being replaced, but it is lost if power is absent for longer than one minute.

The 16-kB ROM contains the basic routines, such as a bootstrap loader that is needed to cold-start the module. It is not possible to run user-written programs directly after a cold start, since the operating system is not present. The built-in software can however activate any connected motors and support the built-in infrared port. The user can see that the module is in this basic mode by the fact that the display shows only an icon representing a standing person, followed by a number between 1 and 5.

The built-in software starts working after it receives support in the form of the RCX operating system, which can be downloaded from a PC. The operating system uses 16 kB of the RAM, so there is 16 kB left for user-developed programs. Since the RCX module utilises a byte-oriented code that is a sort of machine language with around 70 instructions, this is adequate for even the most extensive applications. The RCX operating system is event-oriented and can execute up to six tasks in parallel.

In addition to the H8 processor and the memory chips, the RCX block contains a compact graphic LCD, a miniature loudspeaker (15 mm diameter), an LCD controller, a voltage regulator, two infrared transmitter diodes, a photodiode and a number of passive components.

IR transceiver

The RCX module communicates with the PC via an infrared transmitter/receiver module that is connected to the serial port of the PC with a 9-pin connector. This module is powered by a 9-V battery. Only the TxD, RxD and GND lines of the serial port are used. There is thus no handshaking present.

Figure 5. The RCX module can be completely configured using this screen. The operating system is also downloaded to the RCX module using the menu.

Figure 4. With an extension kit, you can even recreate the well-known R2D2 robot from the Star Wars films in your own living room.

The RTS and CTS pins are shorted together in the IR transceiver. The software can tell whether the module is present by using two of the five interface lines to detect the connection between these two signal lines. There is currently no USB version of the transceiver available, but one presumably will come. Once this version is available, users of Apple Macintosh computers will also be able to work with the Robotics Invention System.

A switch on the front of the module can be used to select a short or long transmission range. In actual use, the module can faultlessly handle a range of several metres.

As soon as the RCX module has been switched on, the infrared transceiver has been connected to a free serial port of the PC and the software on the accompanying CD-ROM has been installed, the system is ready to use. The first action is to download the RCX operating system. This extra step may appear to be inconvenient, but it has the important advantage that the software can be improved and extended in the future. The version supplied with the kit, for example, already incorporates support for the rotary sensor, the temperature sensor and two types of sensors that are not included in the basic kit.

There are by the way other versions of the RCX module available, which are called the Scout and the MicroScout. The Scout is used in the Robotics Discovery Set, which Lego will probably introduce in Europe later this year. This system can only be controlled by a keypad on the Scout module, and is thus more suitable for beginners. It does not come with an infrared interface to the PC. An extension of the Scout module to enable it to be programmed via the PC is presently being

developed, and this possibility is already incorporated into the basic system design. The MicroScout module is used in the Droid Developer Kit, which can be used to build models from the Star Wars films such as R2D2 Astromech Droid. The MicroScout contains seven built-in programs that cannot be modified by the user. The Droid Developer Kit can be used as an extension to the Robotics Invention Kit and is available in Europe.

Getting down to work

Once the operating system has been downloaded into the RCX module, a little man icon appears on the display to indicate that the RCX module is ready for use. Now you can start the software on the PC and enter the realm of the robot designer.

Worked-through examples are provided to guide you through the construction of a number of simple robots. You are then encouraged to develop your own custom software in a step-by-step process. If you should find this too difficult, there are a number of ready-made examples that can be called up at the press of a button. In any case, your first working robot should be ready to go after a couple of hours of construction. After this, adding new functions is just a matter of trying them out and seeing what happens.

In the second part of this mini-series, we will look at how to program your models within the development environment.

Future prospects

Robotics Invention System version 1.5 has been available on the European market for a few months, but it has been available on the American market somewhat longer. If you visit the Lego MindStorms website, you will see that there are two extension kits listed in addition to the basic kit. These kits are called 'Lego MindStorms RoboSports' and 'Lego MindStorms Extreme Creatures'. They contain parts that can be used to further dress up your robots. The 'Lego MindStorms Exploration Mars' kit should also be released in the near future. Complete Mars explorers can be built using this kit. Some extensions to the system, such as an infrared remote control for the robots and the previously mentioned rotary and temperature sensors, are not yet available in Europe. Let's hope that this situation changes quickly!

(000040-1)

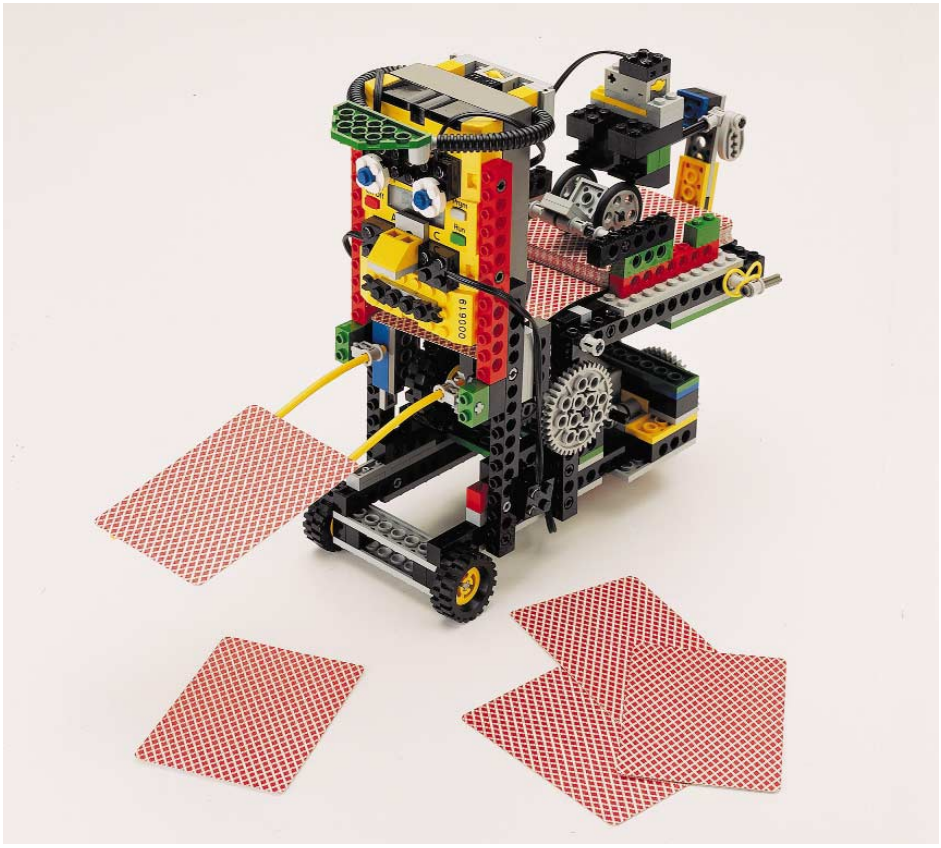


Figure 6. One of the many different robot models that can be built using the basic kit.

Figure 7. Experienced robot builders can give their imaginations free rein with this system. This robot uses no less than three intercommunicating RCX modules and is completely programmed using Visual BASIC.

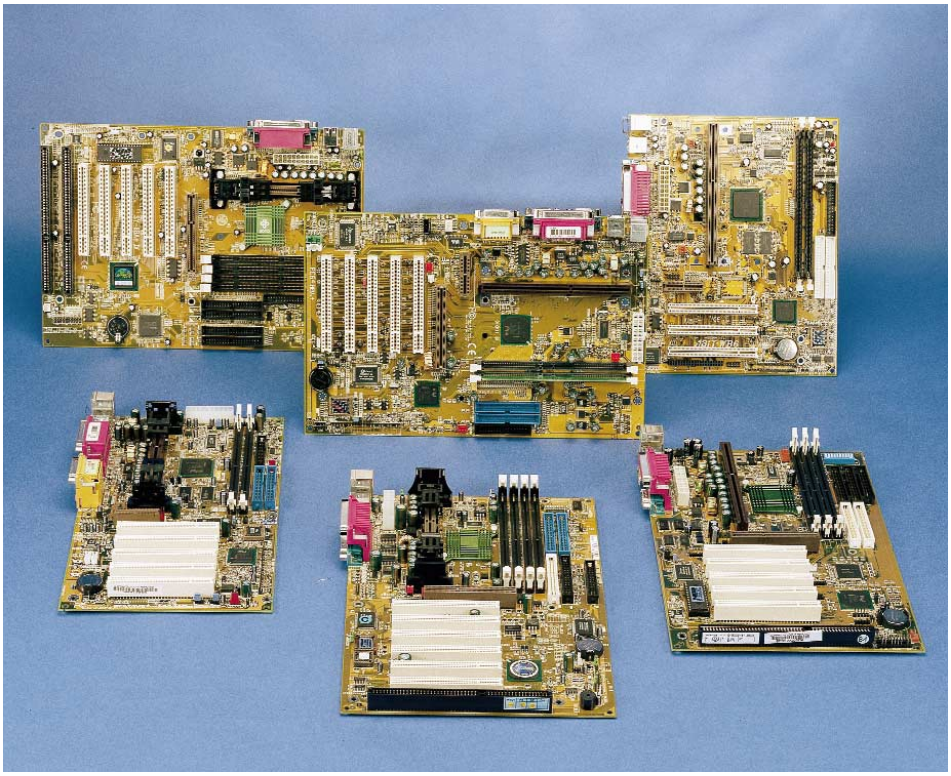
Slot 1 & Slot 2 – Socket 370 – Slot A – Socket 7

PC motherboards

a new generation

By Guy Raedersdorf

It's been nearly two and a half years since we published an article aimed at keeping you informed about the latest developments in the realms of PC motherboards. Given the incredible speed at which PC technology moves these days it seems appropriate to refresh our (and hopefully your) knowledge on the subject.



In the centre, the DFI PC64; next (clockwise), the Abit WB6, Abit BE6, MSI BX Master, DFI PW65E and finally Soyo's SY-6BA-IV.

A motherboard is by definition the foundation on which a PC is built. It combines essential elements like the CPU, chip set, memory, the input/output system and the extension bus (which is usually implemented as a number of connectors).

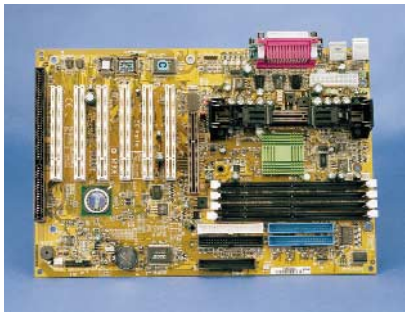
Looking at the virtual explosion of varieties and subvarieties of each of these elements, it is fair to say that the number of combinations is staggering.

Choosing one's own PC motherboard is a bit of a conundrum these days with dozens of manufacturers in the arena, each promoting their own product based on one of many available chip sets, the choice again depending on the CPU type to sit on the motherboard as well as on the features to be offered by the finished product.

Unfortunately, because of the large number of PC motherboard manufacturers it is not possible to give them all a mention in this article. The largest manufacturers are (in alphabetical order): Abit, Aopen, Asus, Biostar, Chaintech, DFI, EliteGroup, Epox, FIC, Gigabyte, Lucky Star, Micro Star International (MSI), NMC, Siemens, Soyo, SuperMicro, TMC and Tyan. There are many more!

This month we will be looking at PC motherboards featuring the **Slot 1** connector (developed for the Pentium II®/III® CPUs; note that some CPU detection software does not distinguish between a Pentium II® and a Pentium III®) and the **Slot 2** connector (developed for the Pentium II/III® Xeon). Note that Soyo's SY-D61GA motherboard is marked by a double slot 2.

One of the first things to remark if we scrutinise the technology applied to the motherboard we came examined for this article is that the majority of manufacturers stick to a BIOS supplied by Award. However, even if a number of manufac-



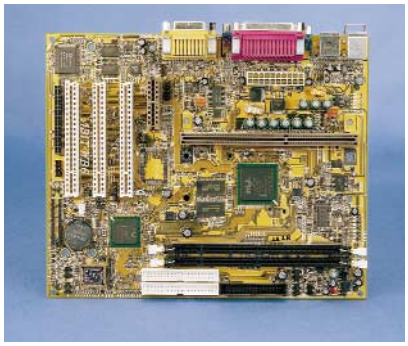
Colour coding to PC99 standard

PS/2 mouse	green
PS/2 keyboard	purple
Parallel port	burgundy
Serial ports	teal or turquoise
Game port	gold
Line Out	lime
Line In	light blue
Mic In	pink
(USB	black)

turers like Gigabyte and Soyo are 100% faithful to their BIOS source, others, including Supermicro and TMC, seem to 'flirt' with other BIOS suppliers like DMI and AMI.

BIOS memory devices like those supplied by Award increasingly feature a write-inhibit function for protection against viruses.

The second fact to note is that it appears to be extremely difficult for a manufacturer to produce a motherboard that is not obsolete the day it is rolled out. The speed of CPUs seems hard to keep up with! For example, as we write this an 800-MHz version of the Pentium III (Athlon®) was announced, so don't be surprised to read about 1-GHz CPUs by the



CPU sockets

Currently, five socket models are known:

1) Slot 1 for the Pentium II® and Pentium III® from Intel as well as for the Celeron®, provided they are mounted on an adapter.
2) Slot 2, it closely resembles Slot 1, but accepts the Pentium II® Xeon only.
3) Slot A designed for AMD's Athlon® CPU.
4) Socket 370 for the Celeron® PPGA from Intel.
5) Socket 7 which accepts a galaxy of CPUs ranging from the K2-6 through the KIII, not forgetting the WinChip and other Cyrix 686's.

Note that there are motherboards with two CPU sockets, for example, Abit's ABP6 which actually accommodates two Celeron® CPUs!

Chip sets

There exists a large number of chip sets from different manufacturers, the best known of which are probably Intel and VIA. Intel currently supplies the types 810 (Slot 1), 810e (Slot 1), 440BX (Slot 1), 440LX, 440ZX (Socket 370), 440GX and 820 (Slot 1). Currently, despite this wide variety, the real battle is between Intel's 440BX, 810(e) and 820 on the one hand, and VIA's competing product called Apollo Pro Plus.
Let's look at some of the more recent chip sets.

810e

This chip set consists of:
– FW82801 (FW = FirmWare), also known as the ICH chip set (I/O Controller Hub). This large-scale integration component (LSI) supports the following functions and is marked by these characteristics:
– complies with PCI Rev 2.2 for functionality up to 33 MHz;
– capable of handling six PCI connectors;
– integrates an IDE controller with Ultra DMA/66 capacity;
– features an USB host interface-for driving two USB ports;
– complies with the AC'97 2.1 standard for audio and telephony coders/decoders (CODECs)
– 82810e GMCH (Graphics Memory Controller Hub). The 82810 also goes by the name of Whitney® (although we must say that the difference in appearance with Ms Houston is striking).
– 82802 Firmware Hub (FWH)
The FWH is a component found in different chip sets from Intel. The hardware characteristics of this component comprise an RNG (Random Number Generator), five GPIs (General Purpose Input), and block latching by register or hardware.

820

After major problems which reportedly have been solved as we write this, the 820 chip set from Intel (a.k.a. Camino®) seems to make good progress, and consequently an increasing number of motherboard manufacturers, including MSI and DFI, are starting to adopt it. The thing about this new chip set is that it supports the new generation of DRAM memory circuits dubbed 'Rambus'.

Destined for use in PCs employing the fastest versions of the Pentium III®, the 820 supports a 4x AGP port, a 1.6-Gbyte/s graphics interface and a data transfer speed of 266 MB/s on the system bus. This chip set comprises a memory controller, an I/O controller and a firmware section consisting of a BIOS stored in Flash memory.

The 820 is important because it enables the new generation of DRDRAM (Direct Rambus Dynamic RAM) components to be employed. In general, the majority of manufacturers provide only two DRDRAM connectors on their motherboards.

What's on a motherboard?

AC97 Audio Controller

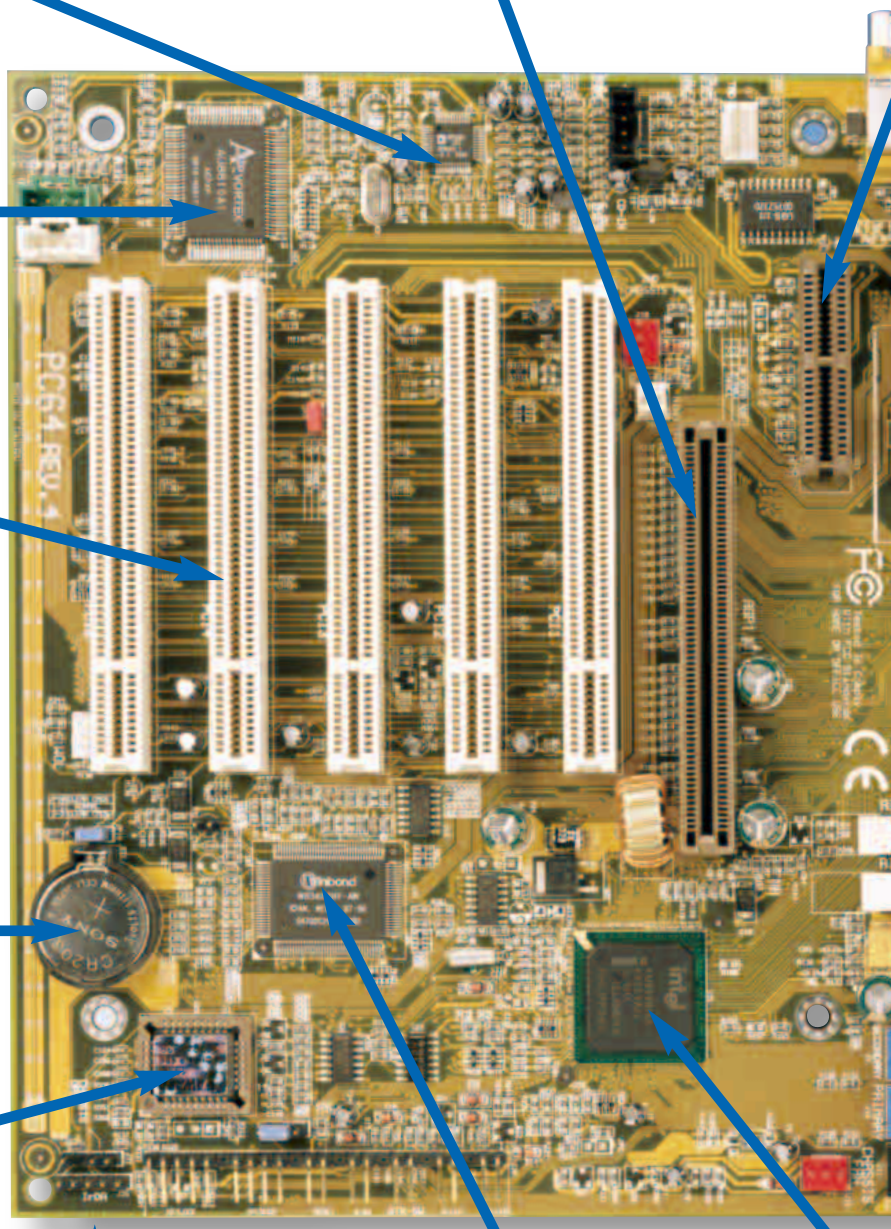
Audio Controller
On this board an Aurela AU8810 is found. An increasing number of motherboards have on-board (integrated) audio and video functions (not on the PC64).

Extension Connectors
These take the form of five PCI slots. Note that ISA slots are not available.

Lithium Backup Battery
This component serves to retain user settings for the system (CMOS Data). Note the presence of a jumper that allows you to return to default data stored in the BIOS ROM (very useful when you can't remember your password).

BIOS
The BIOS acts as a interface between the user system and the electronics on the motherboard. The electronics in turn control various peripheral devices installed in, or connected to, a PC. Nearly all of today's BIOSes are Flash devices, hence the expression 'BIOS re-flashing' for updating the contents of this memory device.
Note that there are motherboards with dual BIOS support (for example, the Abit BP6).

AGP Connector
(Accelerated Graphic Port). This connector receives the video card which is given direct access to the 66-MHz system bus, instead of via the slower (33-MHz) PCI route. Sideband Addressing technology combined with a 2? multiplier allows AGP video cards to communicate with the CPU at a speed of 133 MHz. Today there are also 4-speed AGP connectors with a potential throughput of 1,056 Mbytes/s.



IrDA Connector
By means of this connector, it is possible to establish a wireless (infrared) link between the PC and suitable peripheral devices.

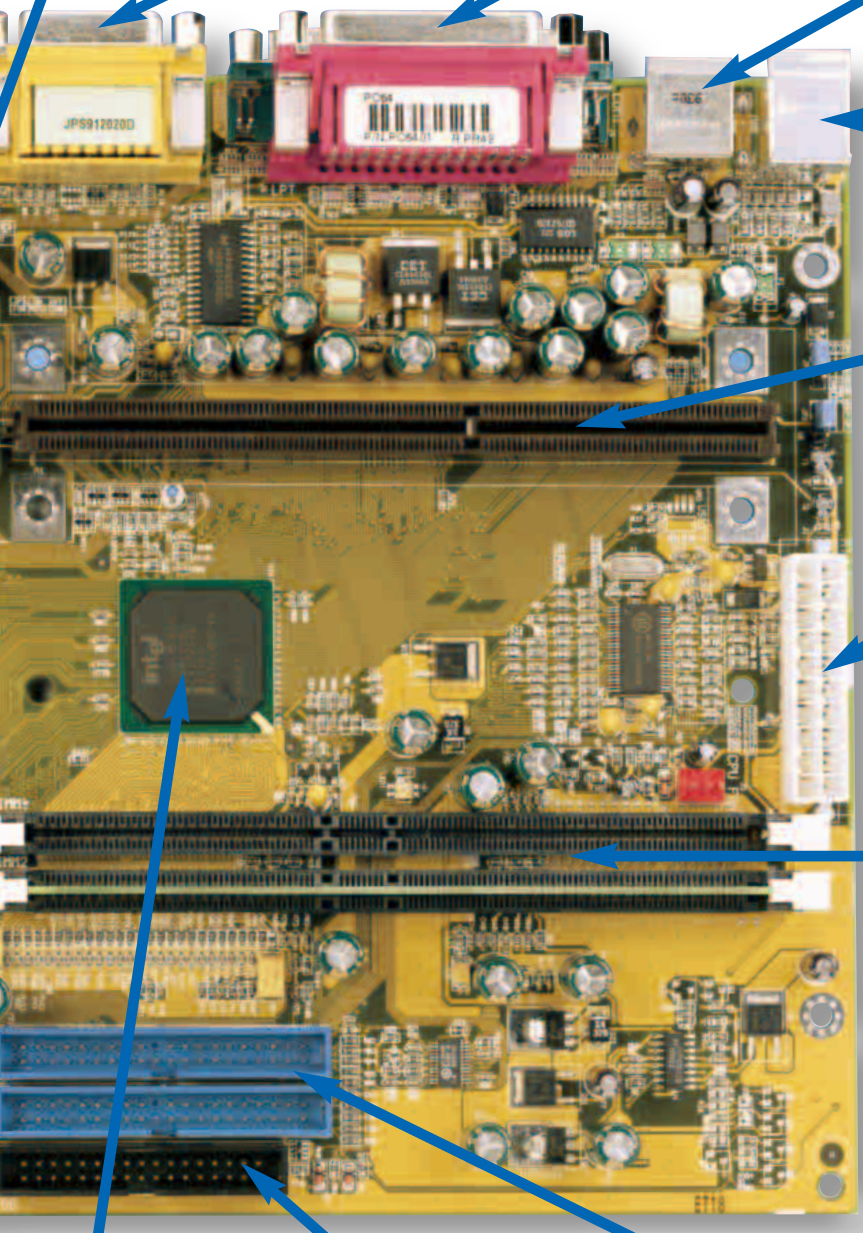
Input/Output (I/O) Hardware

AMR Connector
The AMR (Audio Modem Riser) connector represents an open industry standard which defines an expandable hardware interface supporting audio as well as modem functions.

Game Port Connector and Audio Input/Output Connectors

Parallel Port and COM1-COM2 Connectors

Two USB Ports



PS/2 Mouse and Keyboard Sockets

Slot 1
for the CPU, Pentium II® and Pentium III® in this case, or a Celeron® using an adaptor (see inset on CPU sockets).

ATX Power Connector
A few oddball motherboards have a double supply connector: one 20-pin ATX-style, and one 12-pin Baby-AT style which is even older. Here's proof that the metal case and the power supply are often the only 'components' in a PC to reach a mature age.

Memory connectors
Because of the 820 chip set, the connectors on this motherboard are intended for 184-pin (expensive) SRIMM modules.
However the vast majority of cards currently on the market have connectors for (industry-standard) 168-pin DIMMs. Note that there are motherboards, like SuperMicro's PIIISCA, that allow DIMM (SDRAM) as well as RIMM (RDRAM) to be fitted. 72-pin memory module connectors have virtually disappeared from the scene. A pity because all those 4-Mbyte and 8-Mbyte will go to waste.
Note the absence of a cache module on the board. The CPU now has its own cache, so separate chips are no longer required.

Chip set,
here the 820 is applied, comprising two or three chips.

Floppy Disk Drive Connector (FDD)

Hard Disk, CD-ROM/DVD, IDE1 (primary) and IDE2 (secondary) Connectors

Memory

Once again, new memory devices have appeared on the market, so let's have a look at them.

168-pin DIMM

With the proliferation of no-name 168-pin sockets DIMMs acting as SDRAM in the PC, it has become relatively easy to upgrade your PC with more memory. Since the memory structure is now invariably based on a width of 16 bits, it is no longer necessary to install memory modules in pairs. Today, it is possible to go from 32 Mbytes to 96 Mbytes by the simple addition of one 64-Mbyte DIMM. Meanwhile, you are of course well advised to wait for memory prices to (again) drop to an acceptable level.

RAMbus RIMM

The arrival of the 820 chip set was 'in tune' with the appearance of a new memory device, the RAMbus RIMM.

The maximum available memory depends on the size of each device (chip) used; in fact, it equals 32 times the memory capacity offered by a single chip. Why 32 times? Well, this takes into account that the maximum number of chips supported by a single RAMbus channel is 32, and that the system has just one of these channels. By the way, at 1.6 GB/s, the 'throughput' of a DRDRAM chip is considerably higher than that offered by a traditional DRAM.

At this point, we will have to return to school for some basic arithmetic, or learn to use a calculator. It is sufficient to convert the capacity of the chips in Megabytes to know the total available memory size. Chips with a capacity of 64 Mbits (8 Mbytes) give us a maximum size of 8?32=256 Mbytes. Using 128 Mbit chips this value will double, while 256 Mbit chips offer the highest achievable amount of addressable memory, i.e., 1 GBytes (32?32). But that's not all! The RIMM-style RAM having the largest number of chips must be installed in position 'RIMM1' (rule 1); while the maximum number of chips is 32 (rule 2). If one of the connectors remains empty is has to be fitted with a 'continuity' module (as seen, for instance, on the DFI PC64 motherboard).

VCM-SDRAM memory

Currently, Apollo from VIA is the only chip set to support VCM-SDRAM memory (VCM = Virtual Channel Memory). This modern get-together seems to be marginally faster than other 'couples', like the BX chipset linked to traditional DIMM. VCM-SDRAM modules can be installed in DIMM sockets without any kind of adaptation.

Now who was that again claiming that things would only get simpler?

time this magazine is out (April 2000).

As you will no doubt be able to figure out from the introductory photograph, the following PC motherboard manufacturers (and their distributors) kindly put one or more products to our disposal:
DFI: models PC64 and PW65-E
Abit: models WB6 and BE6
MSI: model BX master
Soyo: model SY-6BA + IV

The co-operation of these suppliers in writing and illustrating this article is gratefully acknowledged.

(000036-1)

Internet addresses

If you want to know more about the motherboards mentioned in this instalment, have a look at the following websites.

DFI : www.dfi.com.tw
Abit : www.abit.com.tw
MSI : www.msi.com.tw
Soyo : www.soyo.nl
Biostar : biostar.com.tw
FIC : fic.com.tw
AOpen : www.aopen.com.tw
Tyan : www.tyan.com
ChainTech :www.chaintech.com.tw
Gigabyte : giga-byte.com.tw

Other interesting links :

Overclocking :
http://hardwarezone.phing.com/
Tom's Hardware :
www.tomshardware.com

Footprints

'ATX' is currently the most popular footprint (or form factor) for a PC motherboard.
Even if called a standard, its dimensions seem to be open to negotiation, ranging from 205mm × 190mm to 305mm × 210mm, while 330mm × 310mm is also encountered.
The older AT footprint has a size of 220mm × 240mm.

Special functions

Suspend to RAM

This function allows the system to be powered down either by pressing the main on/off switch or by selecting the 'Standby' function after closing Windows®98 but without having to close all files, windows and applications to finally be able to close the operating system itself. In this way, you enable the system to store all open program files and data in RAM just before switching off. Once the supply power is restored, the system will be in the same state as before it was switched off.

WOL (Wake-On-Lan)

This connector is used to provide a link to a LAN (Local Area Network) card having a mating socket. In a network, this function allows the system to go into Soft Power Down Mode (Soft-Off). If the system is in Suspend mode, an IRQ or a DMA action is required to put it on the road again.

WOR (Wake-On-Ring)

This connector is usually linked to an internal modem. It allows a system in Suspend or Soft Power Off mode to 'wake up' or start working again (power-on) after detecting an external trigger signal applied to it by a modem.

Chassis Open

When this function is activated in the BIOS, and provided the corresponding detection device actually exists, the system flags an 'open chassis' condition.

RIMM LED

This LED usually lights up red when the system is powered, or when the system is in Suspend mode (Power On Suspend or Suspend to RAM). The same LED goes green (and not with envy) when the system goes into Soft-Off mode.
The same goes for the PCI Standby Power LED which goes red when the PC is in Soft-Off or Suspend mode (Power On Suspend or Suspend to RAM)

active whip aerial

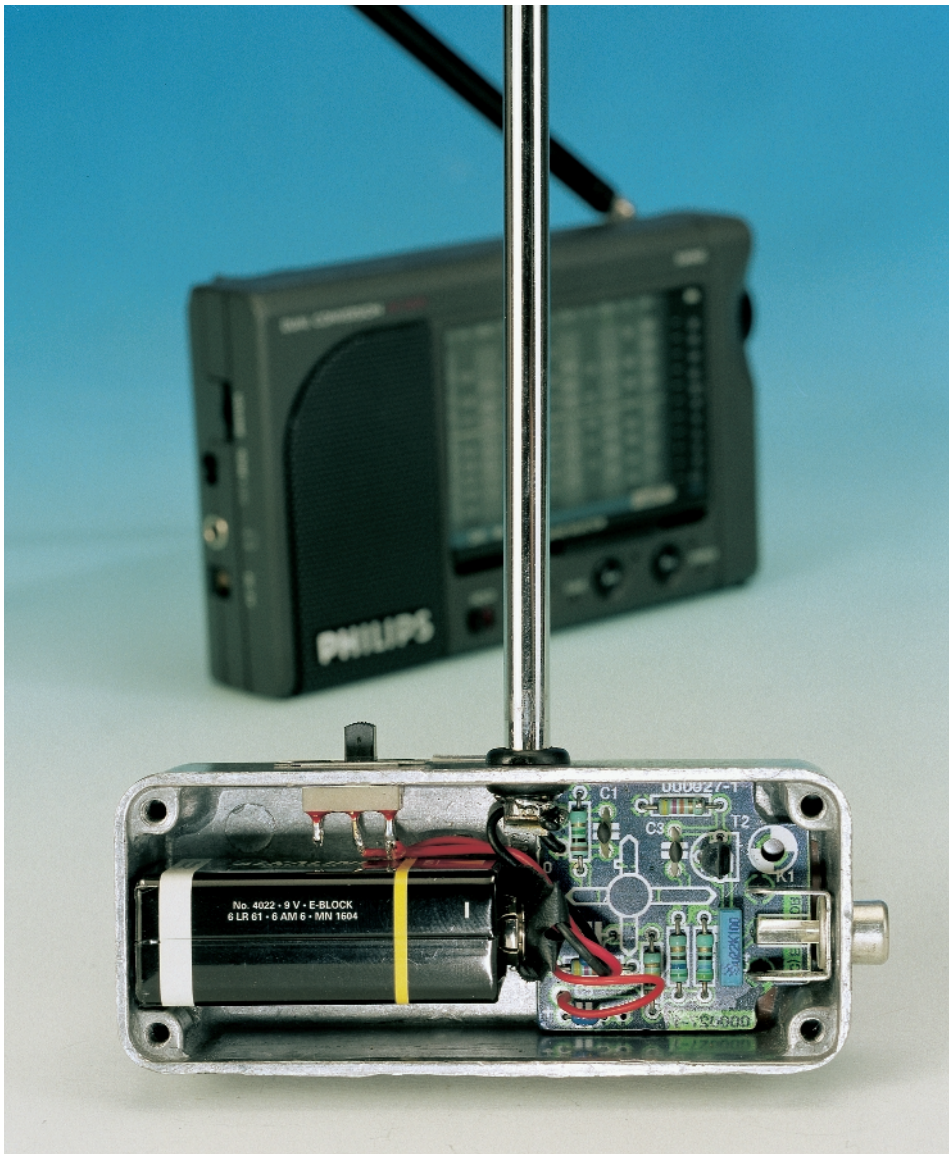
mini short-wave aerial

Design by G. Baars

Short-wave listeners who like to travel a lot have a particular need for an aerial that provides a bit more gain than a standard whip aerial. A second major requirement for a portable aerial is that it ideally should be as compact as possible. This active whip aerial is only 50 cm long, and it completely meets all the demands placed on an ideal travelling companion.

In the January 2000 issue, we already offered short-wave listeners with limited space an alternative to a long-wire aerial, in the form of a simple active loop aerial. This is a very compact aerial, but we fully understand that it is not ideal for use while travelling. You can hardly expect to be able to hammer a few nails into every window frame that you encounter on your journey, just so you can wind a few loops of wire around it. The loop aerial is thus better for stationary use.

What then is the ideal travelling companion for a short-wave listener? If you demand an aerial that provides better reception than a standard whip but is also as compact as possible, you arrive almost automatically at the design described in this article: a short, active whip aerial whose electronics you can fit in your vest pocket, powered by a 9-V battery!



The concept

What most of us would really like to have is a sort of magic aerial: very small and compact, and at the same time much better than its bigger brothers. Unfortunately, magic aerials do not exist, so in real life we always have to accept a compromise. This is not so bad, as long as the result is close enough to what we wanted in the first place.

Of all the known possibilities, the most compact and easy to use type of aerial is without doubt a short whip aerial. If it is made in telescopic form, it need not be

any longer than around ten centimetres when pushed together. It's hard to get any more compact than this.

However, a very short whip aerial has two important drawbacks. Since it is very much shorter than the wavelength of the signals that we want to receive with it (a quarter-lambda whip aerial for the 49-m band would actually have to be 12 m long), its radiation resistance is very high. This creates an immediate difficulty with adapting the aerial to the low-impedance input of the receiver. If we were to couple such an aerial directly to the input of the receiver, this mismatch would mean that only very little of the voltage that the aerial captures from the ether would actually find its way into the receiver.

The second drawback is that a whip aerial with a length of 50 cm (for example) naturally picks up a lot less signal than an aerial that is physically much larger.

These two problems demand an active extension to the aerial. This should at minimum have the following characteristics: (a) high input impedance, (b) low output impedance and (c) a reasonable amount of gain.

With an eye to the high signal levels that are commonly used in the short-wave bands, we must naturally also impose suitable requirements on the large-signal behaviour of the electronics. Finally, if we want to be able to continuously scan the short-wave bands without having to stop to retune the aerial, the active part of the aerial must also have a handsomely wide bandwidth.

All in all, this is a fairly impressive list of requirements!

The realisation

Although our assignment was not exactly easy, if you look at **Figure 1** you can see that the active portion of the aerial is nevertheless a very simple design. It contains only two semiconductor devices and a few resistors and capacitors, nothing more. However, we must remark right away that we intentionally chose to use FETs for the active components, in order to prevent the non-linearity of the amplifier from causing too many problems with intermodulation products. The quadratic input characteristic of a FET gives better results in this regard than the exponential characteristic of a bipolar transistor. In addition, the linearity of the amplifier can be even further improved by operating the FETs with fairly high drain currents.

Let's return to the schematic diagram. The very short whip aerial (50 cm long) is connected to gate 1 of the dual-gate MOSFET transistor T1 via capacitor C1. The unusually high input resistance and low input capacitance of this transistor place almost no load on the aerial. In addition, this MOSFET, which can be used up to very high frequencies, also provides the gain of the amplifier. Since excessive gain does more harm than good, the voltage gain is here limited to around 5 to 10 dB. If desired, the gain can be increased by increasing the voltage on gate 2 (by reducing the value of resistor R2), but doing so causes the problems due to intermodulation and other forms of interference to quickly increase. It is therefore not recommended.

After transistor T1 has taken care of adapting the input impedance and providing the gain, the only problem that we have left is to match the output impedance to the 50-Ω input impedance of the receiver. This job falls to transistor T2, which is a VHF JFET that is biased for a very high drain current so that it can do its work properly. The signal is coupled to the output connector K1 via capacitor C4. The receiver can be connected to K1 via a length of 50-Ω coaxial cable.

The printed circuit board

The printed circuit board is shown in **Figure 2**. It is unfortunately not available ready-made from Readers Services. If you use this circuit board, it is very easy to assemble the circuit, since there are only a handful of components involved. The only detail worth noting is that the dual-gate MOSFET T1 must be soldered to the *bottom side* (copper side) of the circuit board, which is one reason why it is a good idea to pay careful attention to the lead arrangement of this transistor. With the type J310 (or E310) transistor that is used for T2, there are also versions available in an SOT-23 SMD package; these have 'SST' or 'PMBF' as a prefix. The circuit board layout is designed so that these SMD devices will also fit on the board, but in this case T2 must also be soldered to the *bottom side* of the board, just like T1. A J310 in a standard TO-92 package can be mounted in the usual way on the top side (component side) of the board.

A board-mounted female receptacle is used for the output connector K1. The

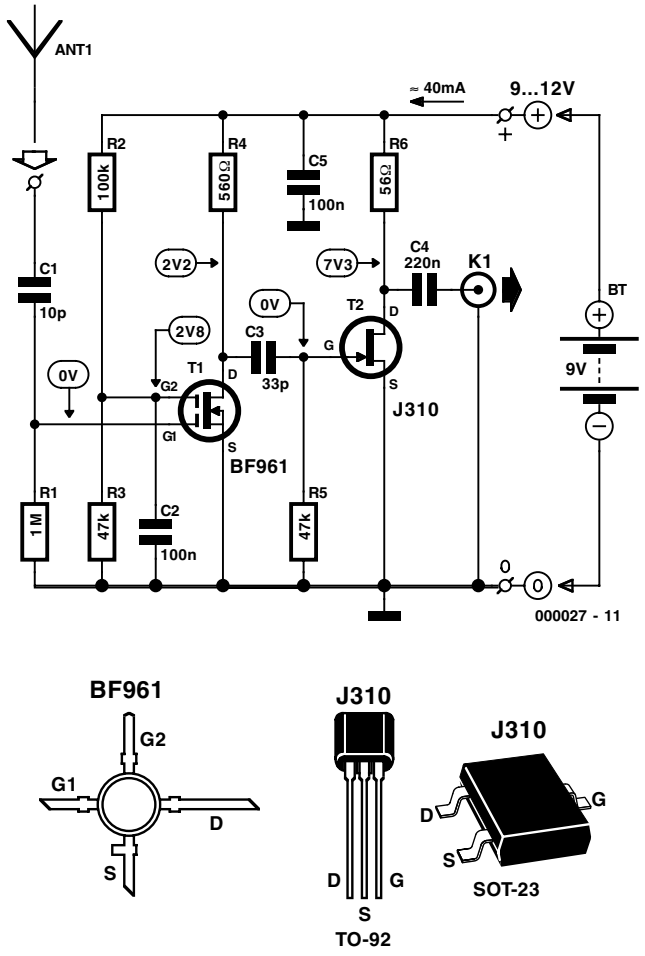


Figure 1. Two FETs provide a proper impedance match for the whip aerial, as well as the necessary gain.

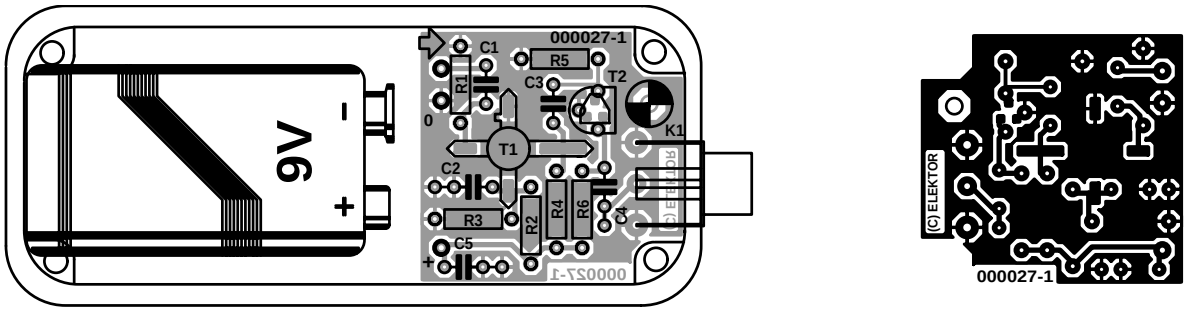


Figure 2. The active circuitry for the antenna can be quickly built using this tiny printed circuit board.

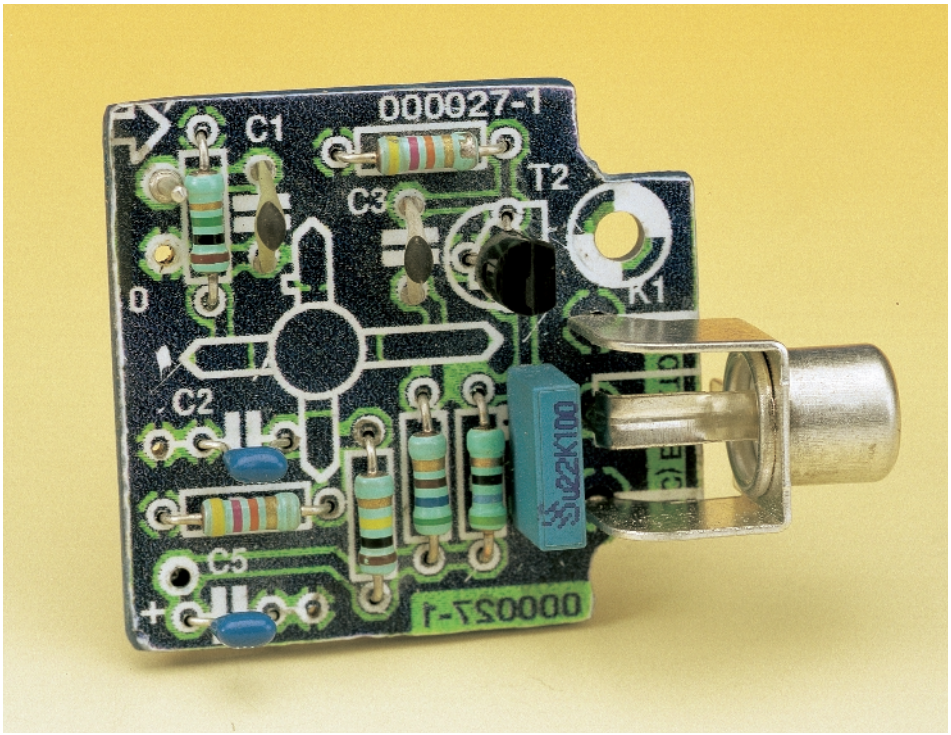
COMPONENTS LIST

Resistors:
R1 = 1MΩ
R2 = 100kΩ
R3,R5 = 47kΩ
R4 = 560Ω
R6 = 56Ω

Capacitors:
C1 = 10pF
C2,C5 = 100nF
C3 = 33pF
C4 = 220nF

Semiconductors:
T1 = BF961
T2 = J310 or E310 (SMD versions: SST310, PMBF310)

Miscellaneous:
K1 = phono (RCA) socket for board mounting
Enclosure: e.g. Hammond type 1590A (90x38x30mm)



manner in which the 50-cm whip aerial is connected to the input pins depends on the particular aerial that is used.

The enclosure

Once you have assembled the circuit board, the first thing you should do is give it a good inspection. After this you can check the voltage measurements marked on the circuit diagram, using a digital voltmeter. These values were measured using a supply voltage of 9 V. The values that you measure may be different, due to the normal tolerance range of the FET characteristics. Deviations from the marked values of up to 25% should be no cause for alarm.

Once everything seems to be in order, it's time to look for a suitable enclosure. A metallic enclosure is preferable. The cast aluminium enclosures made by Hammond are a very good choice. The circuit board together with a 9-V battery will just fit in a type 1590A enclosure. If you use a type 1590B enclosure, you will have room to spare, but this enclosure lacks the charm of the smaller format. There are of course also enclosures available from other manufacturers in which the small circuit board and the battery can be housed in a neat manner.

Conclusion

The current consumption of the circuit is around 40 mA. Since the capacity of modern alkaline batteries is fairly large, a 9-V battery is in principle a very satisfactory source of power when you use the aerial while travelling. For long-term usage, a mains adapter appears to be a more suitable solution. It should have a well regulated output voltage between 9 V and 12 V, with no interference or noise.

There is one final remark. As we noted earlier on, it is not a good idea to choose too high a value for the gain, due to the associated intermodulation products and other forms of distortion. For exactly the same reason, it is also not a good idea to make the whip aerial much longer than 50 cm. With a longer aerial, the increase in the signal strength is marginal, but the distortion level increases quickly.

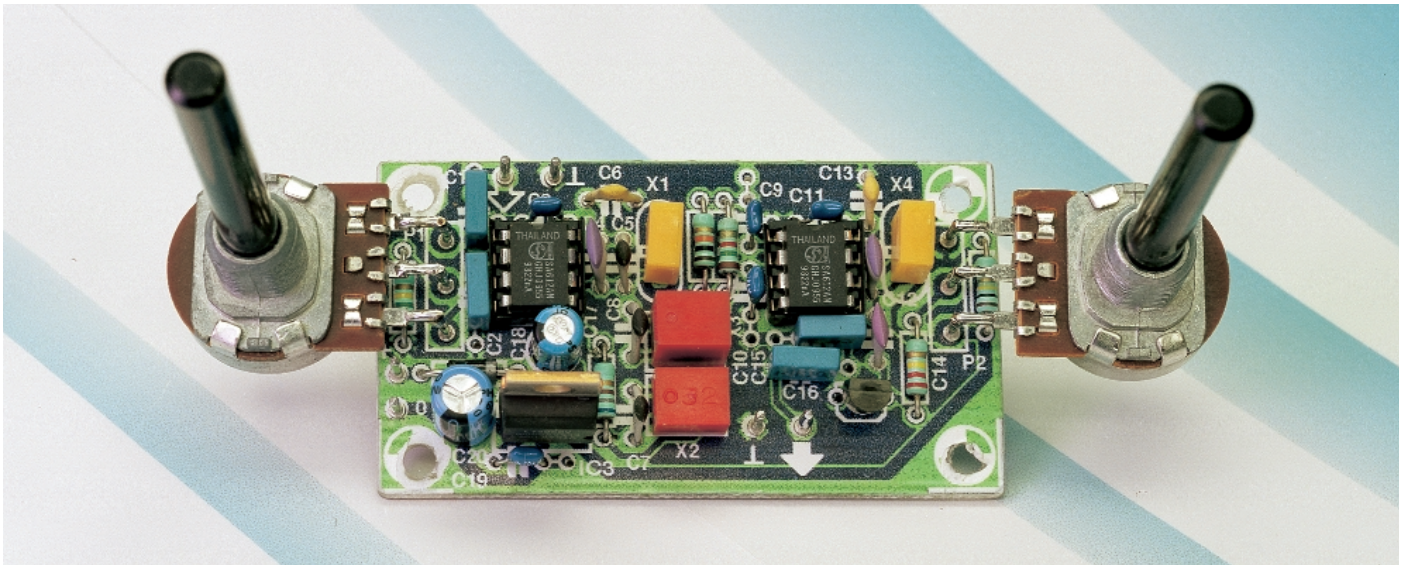
(000027-1)

Design editing: K. Walraven
Text (Dutch original): S. van Rooij

voice shifter

original effects unit

Design by G. Baars



Effects units are and remain popular, especially with pop musicians. These units, phasers, clippers, flangers, and so on are used to make a specific sound or range of sounds. This article describes how such an effects unit may be built from simple means. It is another application of the pitch shifting technique as used in phasing, chorus and flanging, but, unlike true pitch shifting in which a constant interval is created above or below the input signal, it shifts a specific audio band.

It has been said before, and will no doubt be said again many times: we live in a funny world. Most music lovers go out of their way and are prepared to spend a lot of money to achieve music reproduction with as little distortion as possible.

There are, however, others (lovers of a different type of music) who go to great lengths to get away from high fidelity sound. These people feel that the sound processing must match their ideas of what music should be. They try anything and any means to subject voice and instrument sounds to a variety of sound shaping processes that distort the original signal, sometimes by up to 100 per cent.

The voice shifter described here is suitable for seamless connection to a variety of current (commercial) effects units. This is because the applied audio signal is shifted in pitch (frequency). Although this shift does not amount to more than 500 Hz, it is sufficient in practical applications to provide very interesting sound effects with electronic music, but particularly with singing voices. The range of an average voice may be shifted from a rasping male voice to a child's falsetto.

Principle of design

So as to keep the circuit reasonably simple, the design caters for a bandwidth extending from 500 Hz to 6 kHz. This is the

main frequency band for music and also contains the usual human voice register.

The incoming audio signal is combined in a mixer with a carrier at a frequency of 451.5 kHz. As most readers will know, combining two frequencies in a mixer produces output frequencies equal to the sum and difference of the two input frequencies. In the present case, this means bands of 452–458 kHz 445–451 kHz.

As the mixer used is a double-balanced type (see box), the carrier of 451.5 kHz is suppressed. This results in output (side) bands as shown diagrammatically in Figure 1a. Since only one side band is needed (and it does not matter which), a filter is needed to suppress the unwanted band. A further look at Figure 1a shows that thecentre of the righthand sideband is at exactly 455 kHz. That frequency is a well-known one for which there are a plethora of ceramic elements available that lend themselves ideally to producing an excellent, selective 455 kHz band-pass filter.

When the frequency bands in Figure 1a are passed through such a band-pass filter, the residual sideband, that is, the upper one, is shown in Figure 1b.

Two matters remain to be done: the pitch must be shifted and the band of 452–458 kHz must be converted to an audio frequency band. Both may be achieved by applying the residual sideband to a product detector. This is really a kind of mixer in which a carrier wave, variable between 451 kHz and 452 kHz, is added. The result is an audio band 6 kHz wide, which can be shifted by ±500 Hz as shown in Figure 1c. It is this shift that produces the desired effect.

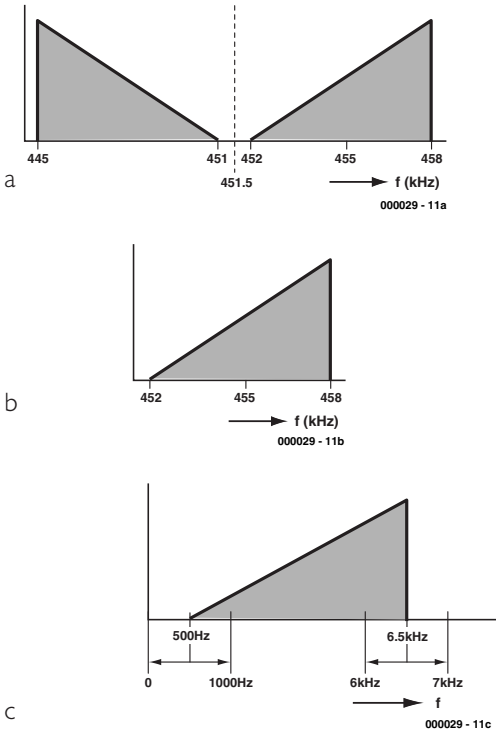


Figure 1. When an audio signal is mixed with a high frequency carrier wave, two sidebands result (1a), of which one, the lower, is suppressed (1b). The residual upper sideband is processed with the aid of a variable beat frequency oscillator (BFO).

Circuit description

Inspection of the diagram in Figure 2 shows that the circuit resembles a typical radio frequency design rather than an audio frequency one.

The circuit is based on two integrated circuits, IC1 and IC2, which are of a type that is normally found only in receivers and transceivers. The same applies to the ceramic filters and resonators. Because of its resembling a radio frequency design, some readers may note that the circuit is very much like that of a single sideband (SSB) exciter followed by an SSB detector with variable beat frequency oscillator, BFO.

The input signal is mixed with the 451.5 kHz oscillator frequency in IC1. The oscillator frequency is generated by elements in the IC and ceramic resonator X1. Note that although the resonant frequency of the resonator is 455 kHz, this is pulled down to the desired 451.5 kHz by capacitors C4–C6.

The filter that suppresses the lower sideband of

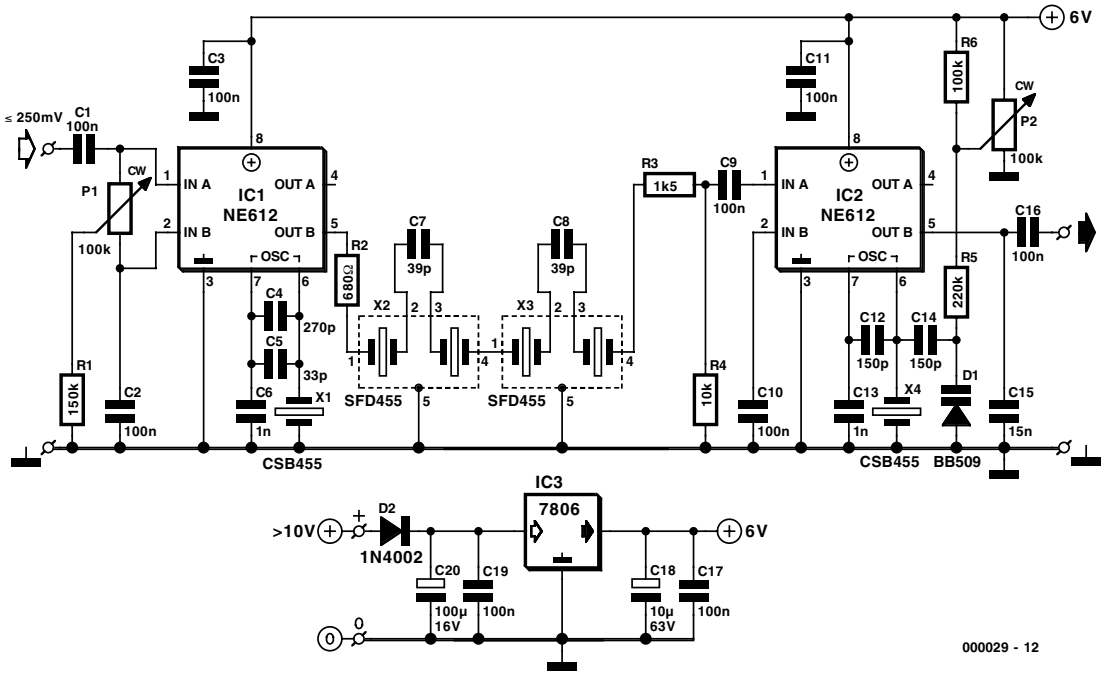


Figure 2. The circuit diagram of the voice shifter shows the simplicity of the design.

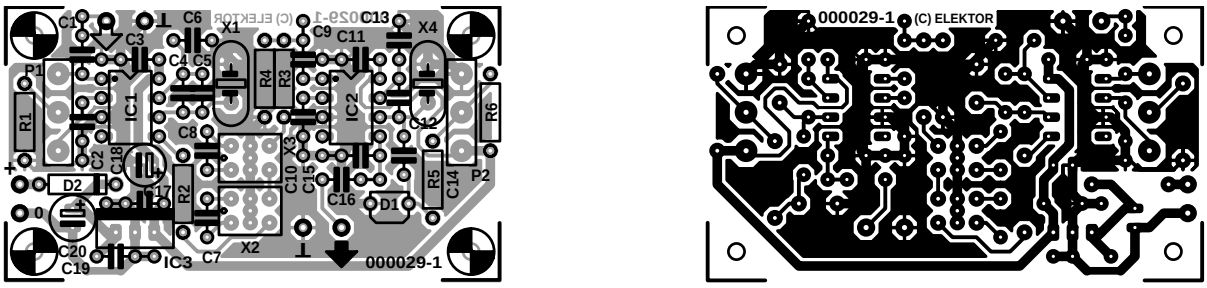


Figure 3. The printed-circuit board for the voice shifter is not available ready made.

the mixer output consists of elements X2 and X3. These are relatively inexpensive items that work very well in this application. Of course, if two small 455 kHz filters are already to hand, they may well do very nicely.

The product detector which, as already mentioned earlier, is a kind of mixer also, is formed by IC2, and is of the same type as IC1. In this case, however, internal elements in conjunction with resonator X4 form a beat frequency oscillator. The frequency of this may be varied with the aid of variable capacitance (varicap) diode D1 and potentiometer P2 over the frequency band 451–452 kHz.

Construction

The voice shifter is best constructed on the printed-circuit board shown in Figure 3. However, this board is not available ready made and must, therefore, be made by the constructor. Fortunately, this is straightforward.

The terminals for input, output, supply voltage, and potentiometers are clearly marked so as to prevent any mistakes.

The reader may also note that in the design of the board account is taken of other, newer types of filter in the X2 and X3 positions, which may have six instead of five pins. There is, therefore, sufficient provision for experimenting with different filter types.

Input and output connections should be made in single screened audio cable.

The supply voltage may be provided by dry or rechargeable batteries, or a mains adaptor. The circuit draws a current of only about 10 mA, so that a battery supply is perfectly all right.

If a mains adaptor is used, bear in mind that the circuit already contains a voltage regulator, IC3, so that the adaptor need not be a regulated type.

The supply voltage must be at least 10 V, but one higher than 12 V is unnecessary. In any case, make sure that the working voltage of C20 is not exceeded.

Setting controls

Network R1-P1 serves to balance the operation of mixer stage IC1, which optimizes the suppression of the carrier wave. Setting P1 to the required position is best carried out by ear. This is done by setting BFO control P2 to one of its extreme positions to make the carrier wave audible as a 500 Hz tone. Balance control P1 is then varied until the loudness of the tone is a minimum.

When P1 is set correctly, the shift effect is controlled with P2. Clearly, there are no rules for this, because it is purely a question of personal taste and requirements.

[000029]

COMPONENTS LIST

Resistors:
R1 = 150kΩ
R2 = 680Ω
R3 = 1kΩ5
R4 = 10kΩ
R5 = 220kΩ
R6 = 100kΩ
P1,P2 = 100kΩ linear potentiometer

Capacitors:
C1,C2,C16 = 100nF MKT (Siemens)
C3,C9,C10,C11,C17,C19 = 100nF ceramic
C4 = 270pF
C5 = 33pF
C6,C13 = 1nF ceramic
C7,C8 = 39pF
C12,C14 = 150pF
C15 = 15nF MKT (Siemens)
C18 = 10µF 63V radial
C20 = 100µF 16V radial

Semiconductors:
D1 = BB509 (ITT)*
D2 = 1N4002
IC1,IC2 = NE612N/SA612AN (Philips)*
IC3 = 7806

Miscellaneous:
X1,X4 = CSB455 (Murata)*
X2,X3 = SFD455A (Murata)*

*) Available from. Barend Hendriksen Elektronica, P.O. Box 66, NL-6970-AB Brummen, The Netherlands. Tel. (+31) 575 561866, fax (+31) 575 565012. Web site: http://www.xs4all.nl/~barendh/

Integrated circuit NE612/SA612

Integrated circuit NE612/SA612 is a double balanced mixer that may be used right up to the very high frequency (VHF) region. It has an internal local oscillator and a voltage regulator.

The device is intended primarily for use in low power communications systems using signal frequencies up to 500 MHz and oscillator frequencies up to 200 MHz.

The mixer proper is a multiplier operating on the Gilbert Cell principle and provides a gain of 14 dB at 49 MHz. The Gilbert Cell is formed by a differential amplifier that drives a balanced switching stage.

The internal local oscillator may be controlled by a quartz crystal, a ceramic resonator, or an LC circuit. It may also be used as a buffer for an external oscillator.

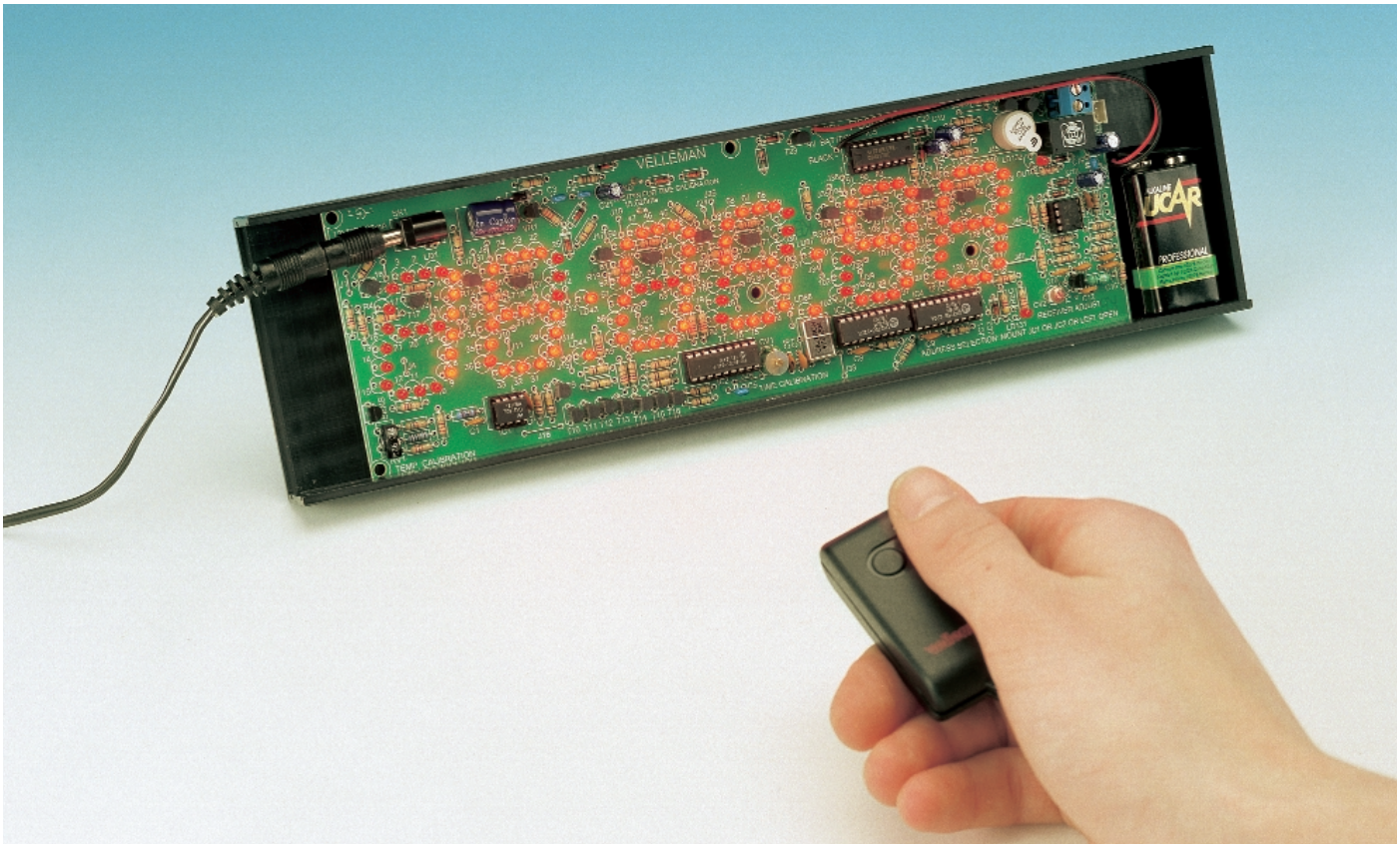
The low current drain makes the device eminently suitable for use in battery powered applications, such as mobile telephones, portable radios, VHF transceivers, and communication receivers.

The device is available in an 8-pin plastic DIL case and as an 8-pin SMD component.

Multifunctional clock

A large display and many features

Thanks to a built-in microprocessor, this clock has a wide range of features and settings. The six large figures of its display, which is made up of many 3-mm LEDs, it can be clearly read at a distance. And you can operate it with a two-button remote control unit.



Features

- 6 large display characters, each 36 mm high
- displays time, date and temperature, either alternately or continuously
- H:M:S: chronometer with lap time indicator
- countdown function to a presettable date, with alarm output
- double scoreboard function (0 – 199)
- random number generator (0 – 99)
- double dice
- mutable acoustical hour signal
- counter function (up or down, 0 – 99)
- relay output for time and temperature alarms
- European or American display format for date, time and temperature
- wireless remote control for all functions
- connections for a backup battery
- temperature range –20 to +70 °C (resolution 1 °C) or 0 to +150 °F (resolution 2 °F)
- power supply 12 V/300 mA (mains adapter)

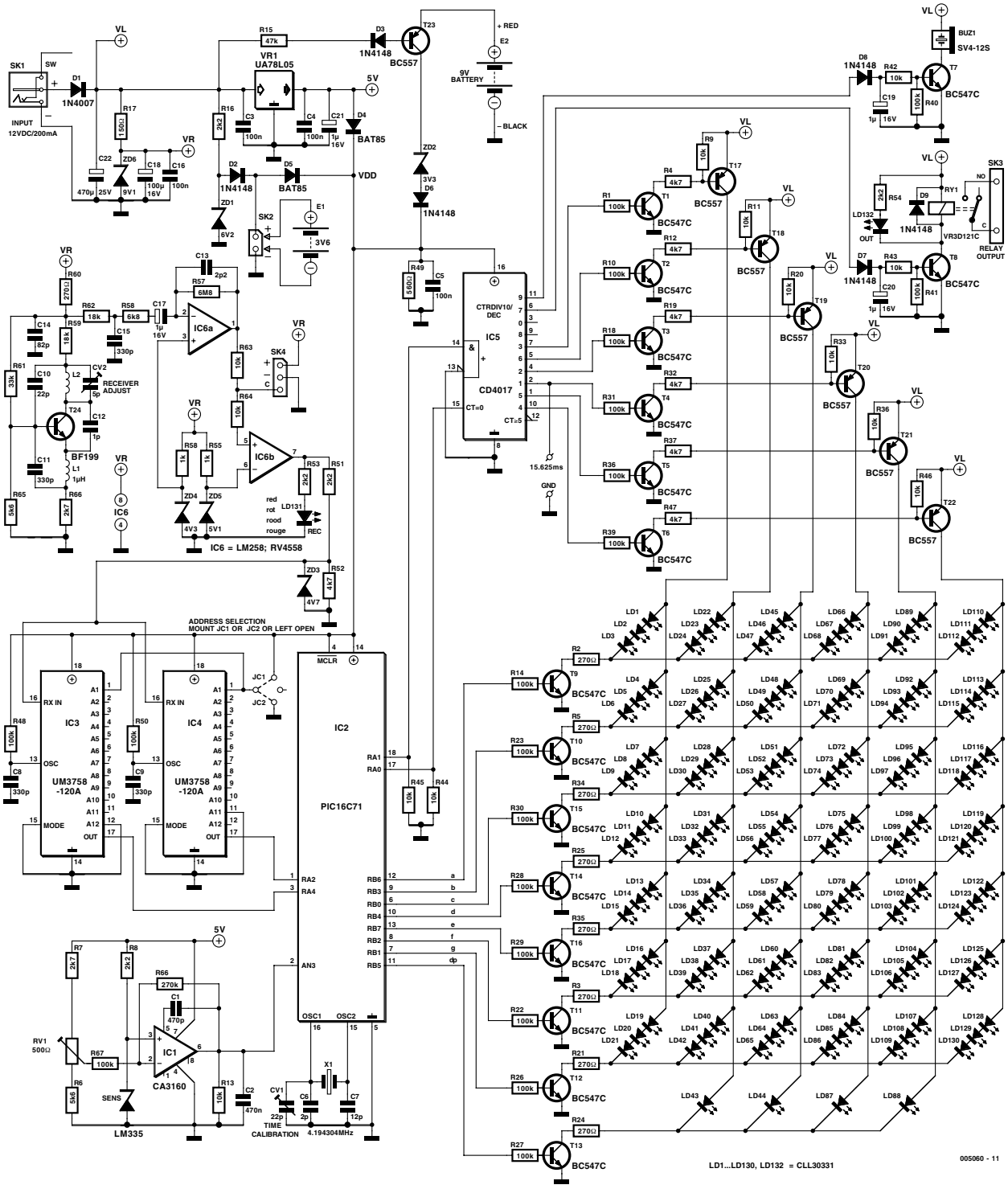


Figure 1. Schematic diagram of the clock, in which a PIC16C715 plays a central role.

Nowadays you can get clocks in all sorts and sizes, from large to small and analogue or digital. Still, the designer of this clock has done his best to produce a unit with number of features that you rarely encounter in a clock.

A few of the more eye-catching characteristics of this design are the very large numerals (36 mm high), which are made up of a number of separate 3-mm LEDs. Not only can the clock display the date and time, it can also display the ambient temperature measured by a sensor. In addition, the clock has an alarm with a switched output, which can be connected to other equipment.

The design

As you can see from **Figure 1**, the heart of the circuit is the PIC16C715 microcontroller (IC2). This device is pre-programmed to drive the LEDs, read in and react to the pushbutton switches, measure the temperature, drive the relay and buzzer and of course keep track of the various clock and timer

states of the circuit.

The LEDs, which are multiplexed, are arranged in a matrix. One side of the matrix is driven by ports RB0 through RB7 of the PIC via transistors T9 through T16, while the other side is driven by ports RA0 and RA1 together with the decade counter IC5 (a 4017), via transistors T1 through T6 and T17 through T22. At each matrix intersection, there are three LEDs connected in series. Together, these form one segment of the display. The LEDs used here are low-current types with a nominal operating current of 2 mA, so that the multiplexed currents remain within reasonable limits.

The receiver/demodulator for the wireless remote control is built around transistor T24 and works in the ISM section of the 433-MHz band. The receiver can be tuned to the frequency of the transmitter (433.92 MHz) by trimmer capacitor CV2. The demodulated signal is buffered and amplified by opamp A1. The output signal is offset by half of the supply voltage level by connecting the + input of A1 to a voltage at this level. Opamp A2 works as a comparator that outputs a blocking voltage if its input signal does not lie above half of the supply voltage level. In this case, the reception indicator LED LC131 does not illuminate. Diode

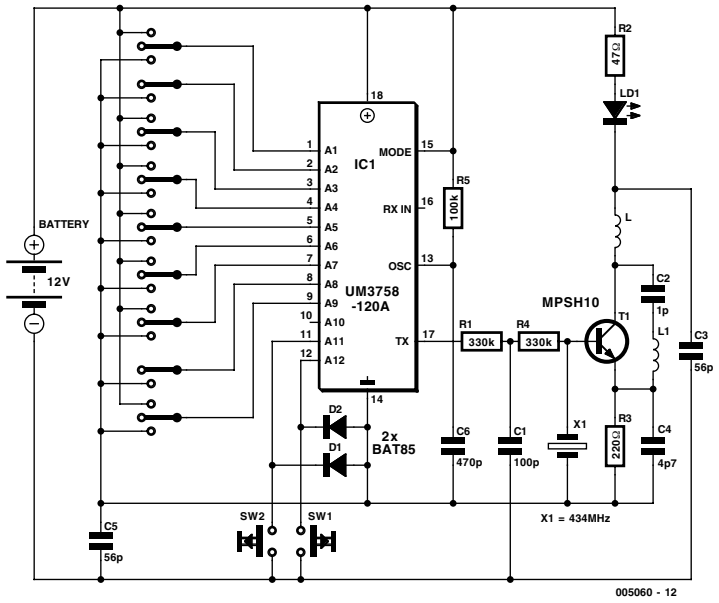


Figure 2. The remote control transmitter can be built in two versions, either with or without wires.

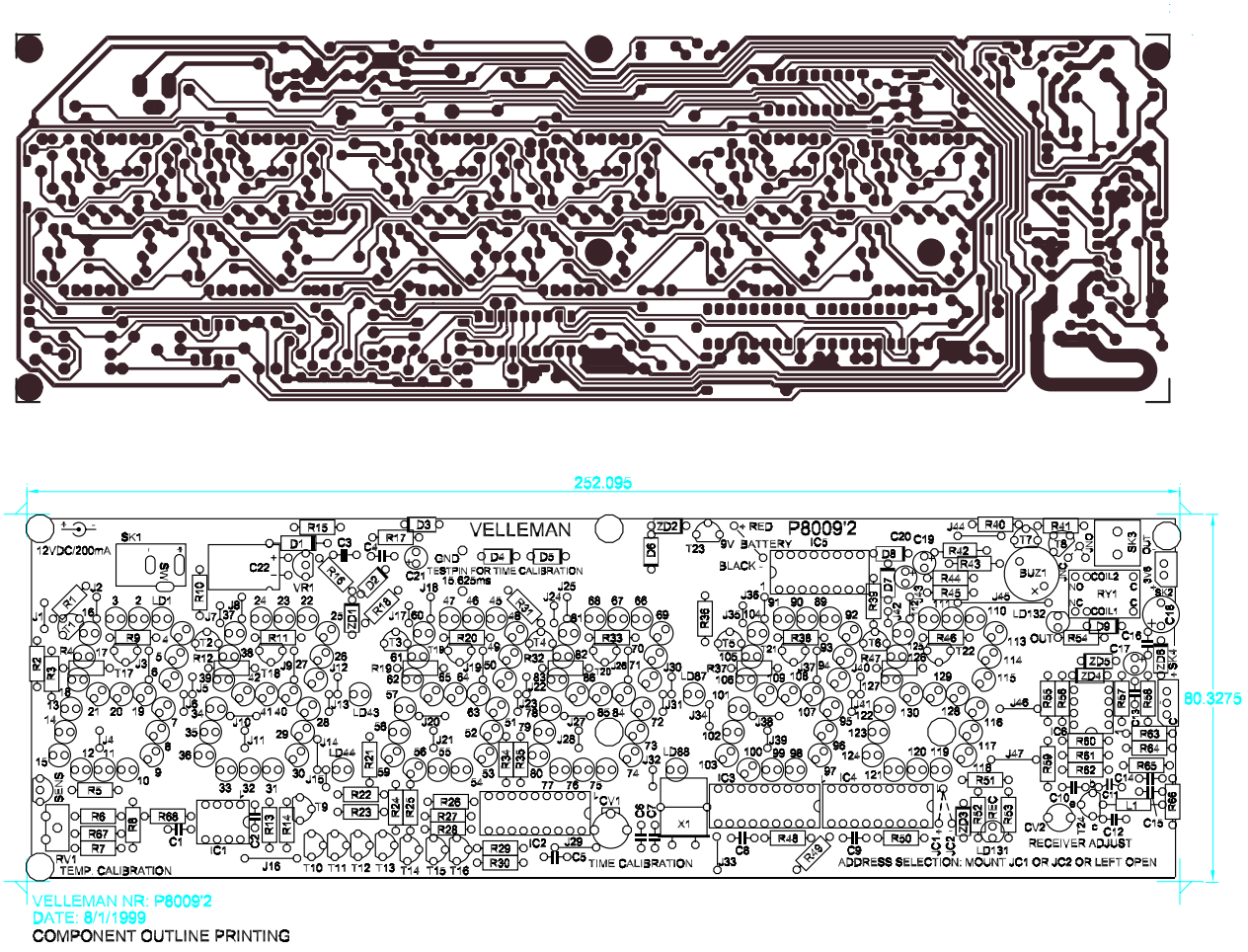


Figure 3. The printed circuit board layout is shown here at 60% of its actual size.

COMPONENTS LIST

Resistors:
R1,R10,R14,R18,R22,R23,R26-R31,R36,R39,R40,R41,R48,R50 = 100kΩ
R2,R3,R5,R21,R24,R25,R34,R35,R60 = 270Ω
R4,R12,R19,R32,R37,R47,R52 = 4kΩ7
R6,R65 = 5kΩ6
R7,R66 = 2kΩ7
R8,R16,R51,R53,R54 = 2kΩ2
R9,R11,R13,R20,R33,R38,R42-R46,R63,R64 = 10kΩ
R15 = 47kΩ
R17 = 150Ω
R49 = 560Ω
R55,R56 = 1kΩ
R57 = 6MΩ8
R58 = 6kΩ8
R59,R62 = 18kΩ
R61 = 33kΩ
R67 = 100kΩ 1%
R68 = 270kΩ 1%
RV1 = 500Ω preset vertical mounting

Inductor:
L1 = 1 µH

Capacitors:
C1,C2 = 470pF
C3,C4,C5,C16 = 100nF
C6,C13 = 2pF2
C7 = 12pF
C8,C9,C11,C15 = 330pF
C10 = 22pF
C12 = 1pF
C14 = 82pF
C17,C19,C21 = 1µF 16V radial
C18 = 100µF 16V radial
C20 = 10µF 16V radial
C22 = 470µF 25V radial
CV1 = 22pF trimmer (green)
CV2 = 5pF5 trimmer

Semiconductors:
D1 = 1N4007
D2,D3,D6-D9 = 1N4148
D4,D5 = BAT85
LD1...LD132 = low-current-LED red, 3 mm (L-934LID)
ZD1 = zener diode 6V2
ZD2 = zener diode 3V3
ZD3 = zener diode 4V7
ZD4 = zener diode 4V3
ZD5 = zener diode 5V1
ZD6 = zener diode 9V1
T1-T16 = BC547C
T17-T23 = BC557B
T24 = BF199
VR1 = 78L05
SENS = LM335
IC1 = CA3160
IC2 = programmed PIC 16C715 (Velleman order code VK8009)
IC3,IC4 = UM3758
IC5 = CD4017
IC6 = LM258

Miscellaneous:
X1 = quartz crystal 4.194304 MHz
BUZ1 = buzzer SV4/12-S (12V DC)
RY1 = relay,12 V, change-over contact (VR3D121C)
SK1 = supply adaptor connector
SK2 = 2-way pinheader
SK3 = 2-way PCB terminal block
Optional: 9-V-battery with clip-on wires
2 solder pins

Transmitter

Resistors:
R1*,R4* = 33kΩ
R2* = 47Ω
R3* = 220Ω
R5 = 100kΩ

Capacitors:
C1* = 100pF, pitch 5 mm
C2* = 1pF, pitch 2.5 mm
C3*,C5* = 56pF
C4* = 4pF7, pitch 2.5 mm
C6 = 470pF, pitch 5mm

Inductor:
L1* = 1 turn (wire)

Semiconductors:
D1,D2 = BAT85
LD1 = LED 3 mm
T1* = MPSH10
IC1 = UM3758

Miscellaneous:
X1* = SAW-resonator type SAW433
SW1,SW2 = miniature pushbutton type KRS0611
2 battery contacts
12-V battery

** only required for wireless version*

ZD3 limits this signal so that it cannot exceed the supply voltage level of the following ICs (IC3 and IC4). Each of these two ICs decodes one of the two pushbutton switches of the remote control unit and activates the microcontroller's RA4 or RA3 input respectively. Jumpers JC1 and JC2 can be used to set your own personal code.

The temperature is measured using the well-known LM355 sensor from National Semiconductor. IC1 compares the output voltage of the sensor to a reference voltage provided by the voltage divider formed by R7, RV1 and R6. The trimmer potentiometer RV1 is used to calibrate the measurement. The DC voltage from IC1 is passed to an 8-bit A/D converter contained in the PIC microcontroller.

The power supply looks a bit complicated, since there are two different options for backup power for the clock. You can choose between a rechargeable NiCd battery (E1) and a 9-V battery (E2).

Diode D1 protects the circuit against a reverse-polarity connection of the mains adapter. The voltage at the cathode of D1 is used directly to supply the LED matrix, the buzzer and the relay circuit. This voltage is also reduced to 9 V by the combination of R17 and ZD6, and this lower voltage is used to power the receiver stage. The voltage on the other side of D1 is used to charge the optional NiCd battery E1 via resistor R16.

Diodes D5, D4 and D6, together with the circuitry around transistor T23, ensure that the NiCd battery or the 9-V battery supplies power to the circuit if the voltage from the adapter is not present.

Make sure that the mains adapter can provide a voltage of at least 12 V DC. If the voltage is lower than this, problems may occur in the remote control receiver circuit.

Transistor T7 drives the buzzer, while T8 switches the relay and the associated indicator LED LD132.

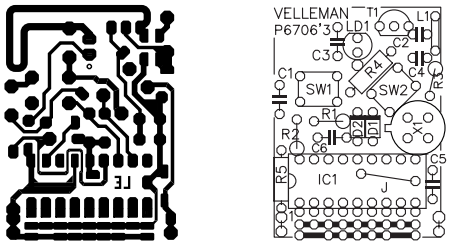
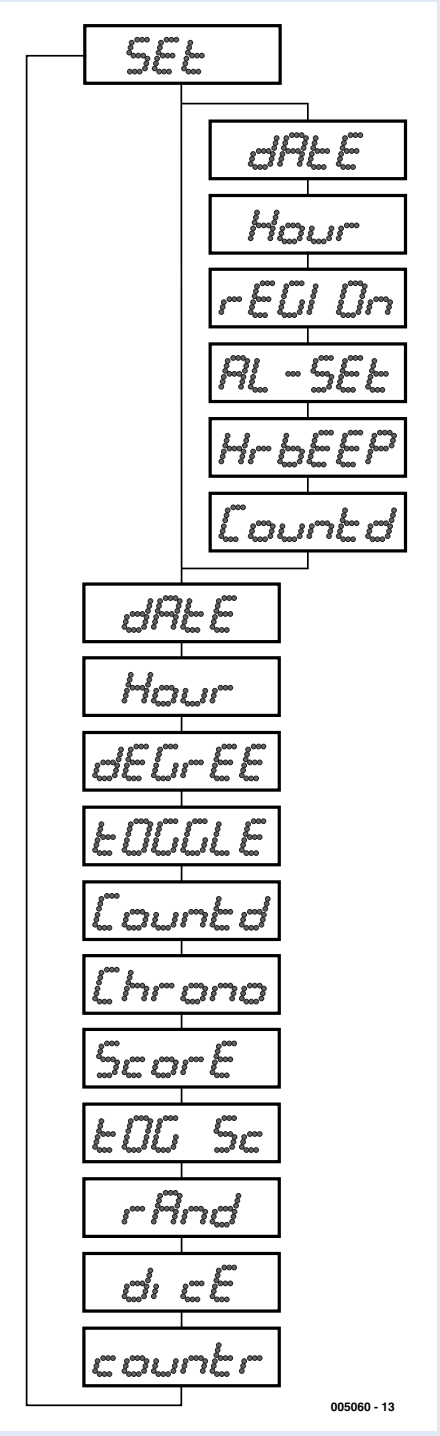


Figure 4. The remote control transmitter can be built on this printed circuit board.

Operation

The clock and all of its functions are operated by a single control unit with only two pushbuttons, which are called 'button 1' and 'button 2' or the left and right buttons, respectively. Clearly, this means that a menu structure must be used to make it possible to control all the functions of the clock (see **Figure A**).



SET
This menu item is used to set the date (DATE) and time (HOUR), to select American or European date format (mm/dd/yy or dd/mm/yy, respectively) and Fahrenheit or Celsius units for the temperature display, and to set the alarm. Two separate submenus allow you to specify whether the clock should emit an acoustic signal on each hour and to set the interval for the countdown timer.
The alarm can be triggered in two different ways. The first is when a specific time is reached (for which no date setting is possible), and the second is when a specific temperature is exceeded. The relay is activated when the alarm is triggered.
Use button 1 to scroll through the menu and button 2 to select a menu item.

– DATE
Button 1 increases the value of the blinking display element, and button 2 selects the next element.

– HOUR
This is set in the same manner as the date. The seconds display is always held at zero. The clock starts to run when button 2 is pressed.

– REGION
Button 1 alternately selects American or European display formats (default is European). The American display format is: 12-hour clock, month/day/year, Fahrenheit degrees; the European format is: 24-hour clock, day/month/year, Celsius degrees.

– AL-SET
Button 1 enables or disables the alarm; the selection must be confirmed by pressing button 2. If the alarm is enabled, button 1 can then be used to select between time and temperature as the trigger source, and this must again be confirmed by pressing button 2. The alarm time is set in the same manner as the time of day. The alarm temperature setting can be increased by pressing button 1, and the setting can be confirmed by pressing button 2.

– HRBEEP
Button 1 enables or disables the hour beep. The selection must be confirmed by pressing button 2.

– COUNTD
The date for the countdown function is set here, in the same manner as with DATE.

All of the following setup menus are exited by pressing button 1.

DATE
Display the date.

HOUR
Display the time.

DEGREE
Display the temperature in Fahrenheit or Celsius degrees, depending on the selected date and time format.

TOGGLE
If this option is selected, the time, date and temperate are displayed in alternating sequence (the date and temperature are displayed for a shorter interval than the time).

COUNTD
This menu item is used to display the remaining hours, minutes and seconds until the date set using the SET menu. Once this time has expired, the relay is enabled and a blinking '00:00:00' is displayed. If either of the control buttons is pressed, the relay is disengaged and the normal time display reappears.

CHRONO
This is a stopwatch function with a resolution of one second. Button 2 starts and stops the chronometer. Pressing button 1 stores a lap time, which is displayed if button 1 is pressed again. If button 1 is pressed and held for longer than three seconds while the chronometer is stopped, it is reset.

SCORE
This menu item displays a double scoreboard. Button 1 increases the left-hand score, while button 2 increases the right-hand score. If either one of the buttons is pressed and held for three seconds, the associated score is reduced. If button 2 is pressed and held for longer than three seconds, both scores are reset.

TOGSC
This causes the time and the scores to be displayed alternately.

RAND
The two middle digits display a rapidly changing series of random numbers as long as button 2 is held pressed.

DICE
Each of two digits displays a random number from 1 to 6. Both 'dice' are thrown at the same time.

COUNTR
This is a count-up and count-down function. Pressing button 2 increments the count by 1, while pressing button 1 decrements the count.

The schematic diagram of the remote control transmitter in Figure 2 shows that the same type of encoder IC is used in the transmitter as in the clock (IC1). The transmitter can be set to the same code as the receiver by means of the two jumpers.

If you want to build the wireless version of the control unit, the oscillator portion around T1 is necessary. The components list indicates which additional parts must be installed for the wireless version. With the wired version, three wires must be connected between the +, – and TX points on the transmitter board and the +, – and C pins of connector SK4 on the clock circuit board, respectively.

Construction

An experienced solder artist should have no problems constructing the circuit with the help of the printed circuit board track and component layouts shown in Figure 3. Those of you with less experience will find the kit that can be obtained from Velleman helpful, since it contains an extensive set of instructions. As long as you carefully follow these instructions, nothing can go wrong. Be sure to mark each component on the checklist included in the construction guide once you have mounted it on the circuit board. In addition, all the components in the kit are arranged in the order in which they should be inserted in the board, including the jumpers, which certainly makes it easier to assemble the circuit. Pay close attention to where each component should be placed on the board, since this is not always easy to see in places where the components are close together. In such cases the schematic diagram and the photograph of the board can help. Finally, note what jumpers you have installed for coding the remote control receiver on the clock board (none, JC1 or JC2), and make sure to use the same setting for the remote control unit.

Alignment

There are three set-up points on the main circuit board. Preset RV1 adjusts the temperature indication, trimmer capacitor CV2 tunes the receiver for the remote control and trimmer CV1 adjusts the clock itself.
Once the 12-V mains adapter has been connected, the displays should all start to blink. After this, the receiver for the remote control can be adjusted (assuming you have built the wireless version). Hold the transmitter around one metre away from the clock board and press on one of the buttons. Rotate trimmer CV2 with a non-metallic screwdriver until the reception LED starts to blink. Then repeat this procedure with the transmitter further away (up to 10 m).

You will need an accurate thermometer to calibrate the temperature indication. Place the thermometer next to the temperature sensor, wait a little while and then choose the 'degree' function in the menu. This causes the temperature to be displayed. Rotate RV1 until the displayed temperature matches the reading of the thermometer. To make sure that the calibration is OK, you can wait a while longer, until the entire circuit is fully warmed up, and then check if the temperature indication is still correct.

Finally, you have to calibrate the time base. Set CV1 to the middle of its range and check periodically (such as once an hour) how fast or how slow the clock is running. If the clock runs too fast, turn CV1 more towards its maximum value, and if it runs too slow turn CV1 in the other direction. If you don't want to wait so long and you have a frequency counter, you can also measure the frequency or period of the signal between TESTPIN and GND. These are located on the circuit board above the third display digit from the left. Set CV1 for a frequency of 64 Hz or a period of 15.625 ms.

The clock is now ready for use, and the circuit board can be installed in a suitable enclosure.

(005060-1)

A complete construction kit for the clock can be obtained from Velleman, Belgium. This includes the printed circuit boards, pre-programmed PIC and all other components. It priced at US$89.95 and available from electronics retail shops (part number K8009). An enclosure is separately available (part number B8009). The pre-programmed PIC is also available by itself from Velleman (part number VKB009). In the UK, a selection of Velleman kits is stocked by Maplin.

Velleman contact information:
Velleman Components,
Legen Heirweg 33, B-9890 Gavere, Belgium.
Fax: (+32) 9 384 6702.
Email: order@velleman.be
Internet: http://www.velleman.be

Simple high-side current meter

By G. Kleine

Maxim has recently introduced a new integrated current meter IC called the MAX4173. It converts a current measured in the positive supply lead into a proportional voltage to earth. This can in turn be input into the A/D converter of a system.

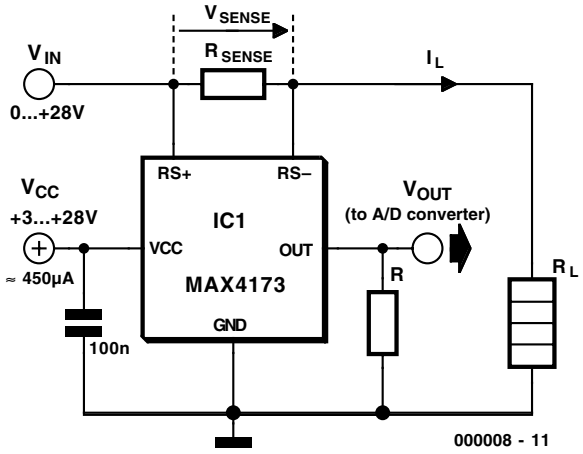


Figure 1. Basic application circuit for the MAX4173.

The MAX4173 allows a control microprocessor to measure and monitor the current consumption of electronic components. The main advantage of this new IC is that the voltage V_{in} on the supply lead to be measured may be higher (as well as lower) than the supply voltage V_{CC} of the IC itself. This means for example that it is no problem to use a MAX4173 to measure the current on a +12-V line, even if the IC itself is connected to the +5-V or +3.3-V supply line of the processor. It is not necessary for both supply voltages to be the same, as with previous designs.

The IC can work with a supply voltage between +3 V and +28 V, and it draws only around 450 µA of current. The voltage V_{in} in the measurement arm may lie between 0 and +28 V, independent of the operating voltage of the IC. This makes this IC an outstanding choice for battery charging control systems, among others. However, the measurement error increases greatly if V_{in} is less than 1 V.

Figure 1 shows the basic connection diagram for the MAX4173 current measurement IC. The most important external element is the sense resistor R_{sense}, which is located in the current path. Resistor R_L represents the connected load, which can be another circuit or a rechargeable battery, for example. The output voltage V_{out} must have a very high input impedance load. This voltage is proportional to the magnitude of the current through R_{sense}. Later on we will see how to calculate the value of R_{sense}.

Internal design of the current sensor IC

The MAX4173 consists of an operational amplifier, a transistor and a current mirror, as shown in Figure 2. If we assume that the operational amplifier is an ideal amplifier, then no current flows in the inverting input. The voltage at this input is thus equal to that at RS–. This means that the voltage across R_{R1} is equal to that across R_{sense}, since the transistor closes the feedback loop around the operational amplifier. The result is that the current in resistor R_{G1} is lower than the load current I_L by a factor of R_{sense} / R_{G1}:

$$I_{RG1} = I_L \cdot R_{sense} / R_{G1}$$

This current reaches the current mirror, which converts it into another current

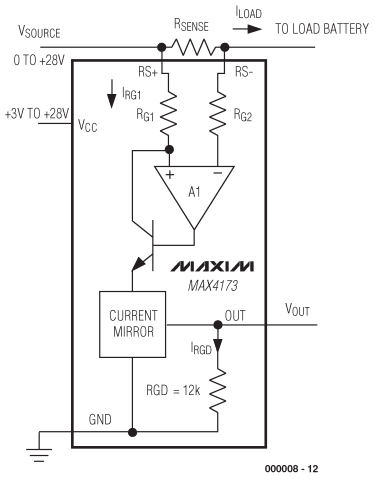


Figure 2. Internal schematic diagram of the MAX4173.

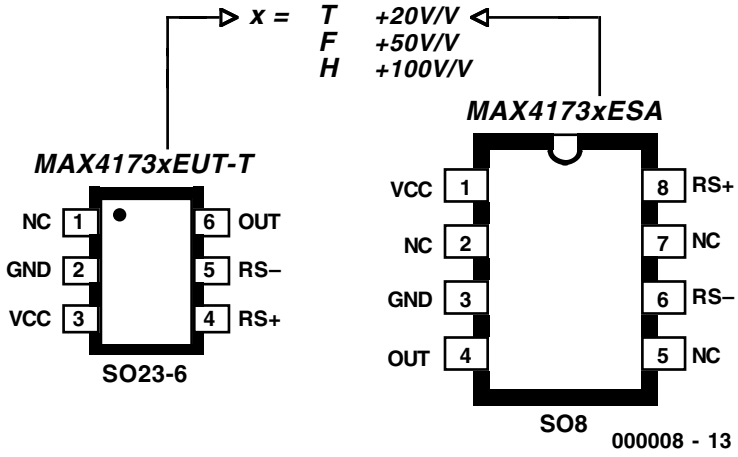


Figure 3. Pin assignments for both package types.

I_{RGD} with an amplification factor B:

$I_{RGD} = B\ I_{RG1}$

The internal 12-kΩ resistor R_{GD} converts I_{RGD} into the output voltage V_{out}, which can be picked up by a high-impedance circuit.

The conversion gain G of the current sensor can be defined as:

$G = V_{out} / V_{sense} = B \cdot R_{GD}/R_{G1}$

The transfer function of the IC from the load current I_L to the output voltage V_{out} is thus:

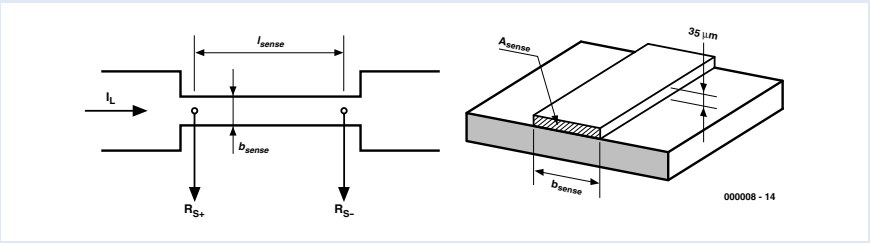
$V_{out} = G\ I_L\ R_{sense}$

A prerequisite for the accurate operation of the IC is that the output voltage V_{out} must not be connected to a low-impedance load. To minimise the error, the load resistance of the following circuit stage (A/D converter) should be greater than 100 kΩ. For best results, a high-impedance voltage follower should be used as an output buffer.

Circuit board tracks as sense resistors

Current-carrying capacities of printed circuit tracks, for dimensioning a printed circuit track used as a sense resistor.

Track width b_{sense}	Allowed current I_{sense}
b	I
0.1 mm	0.5 A
0.2 mm	0.7 A
0.3 mm	1.0 A
0.5 mm	1.5 A
0.8 mm	2.5 A
1.0 mm	3.5 A
1.5 mm	5.0 A
2 mm	7.0 A
3 mm	8.5 A
5 mm	12 A
10 mm	20 A



The sense resistance can also be produced using a section of a thin copper circuit board track. The specific resistance of copper is

$\rho_{Cu} = 0.0175\ \Omega\ mm^2 / m.$

With standard printed circuit boards, the thickness of the copper lamination is 35 µm. The value of R_{sense} can be calculated as

$R_{sense} = \rho_{Cu}\ l_{sense} / A_{sense}$

where

$A_{sense} = 35\ \mu m\ b_{sense}$

For example, suppose that we need an R_{sense} of 100 mΩ and the track width is 0.2 mm. If we rearrange the formula for R_{sense} to solve for l_{sense} and insert these values, we find that the value of l_{sense} is 4 mm. We also have to take into account that the copper track must be able to handle the power that is dissipated. The values in the table can be used as a guideline for selecting the width of the copper track for R_{sense}. These are based on an assumed temperature rise of 30 °C.
The last point to consider is that copper has a relatively high thermal coefficient of resistance. The resistance changes by 0.4% per °C, which amounts to no less than 16% over an ambient temperature range of +5 to +45 °C.

MAX4173 versions

The MAX4173 is available with three different gain classes and two different package types. In addition to the usual SO-8 package, there is also an especially small SOT-23 package with six leads (see Figure 3). The letter at the end of the part number indicates the gain from V_{sense} to V_{out}, as shown in Table 1.

Table 2 shows the recommended component values for various load currents. The value of R_{sense} is chosen such that it drops 100 mV at the maximum level of the load current I_L. The maximum allowed voltage difference between the sensor inputs RS+ and RS– is 300 mV.

One thing that you have to watch out for is that the operating voltage V_{CC} of the MAX4173 (but not the voltage V_{in}) must be at least 1.2 V higher than the maximum level of the output voltage $V_{out,max}$. Otherwise, the output voltage V_{out} will not remain proportional to the measured current at high current levels.

The output voltage is generated by an internal current source that feeds current to a 12-kΩ resistor connected to earth (see Figure 2). This is why a low-impedance load should not be connected to the output. The measurement error due to the load impedance R_{out} can be calculated using the formula

$$\text{Error},\% = 100\ \% \times [(\text{Rout} / (12\ \text{k}\Omega + R_{out}))\text{-}1]$$

This means that even with a load resistance of 100 kΩ, the error is 10%, and it is still 2% at 500 kΩ. The specified measurement error of the MAX4173 is ±0.5% under normal operating conditions, with the voltage V_{in} at RS+ greater than 2 V. The measurement error increases as this voltage drops below 2 V. At 0.1 V it is typically 9%.

Consequently, if the A/D converter following the current sensor needs to be driven by a low-impedance source, an operational amplifier with a high-impedance input must be used as a buffer between the sensor IC and the A/D converter. This amplifier must allow input and output voltages going down to earth level with a single supply voltage.

Calculating R_{SENSE}

The value of the sense resistor R_{sense} can be calculated using the formula

$$R_{sense} = V_{sense} / I_{Lmax}$$

The nominal value of V_{sense} is taken to be 100 mV at the maximum load current. For linear operation, the voltage between RS+ and RS– must be less than 300 mV. After calculating the value of R_{sense}, you can choose a suitable standard resistance value or a combination of two or three standard values to make up the sense resistor. Don't forget that the resistor you have chosen (or the combination of resistors connected in parallel) will dissipate an amount of power given by

$$P_{v,max} = R_{sense}\ I_{Lmax}^2$$

which it or they must be able to convert into heat.

(000008-1)

Literature:
MAX4173 data sheet, available from http://www.maxim-ic.com.

Table 1. MAX4173 gain options.

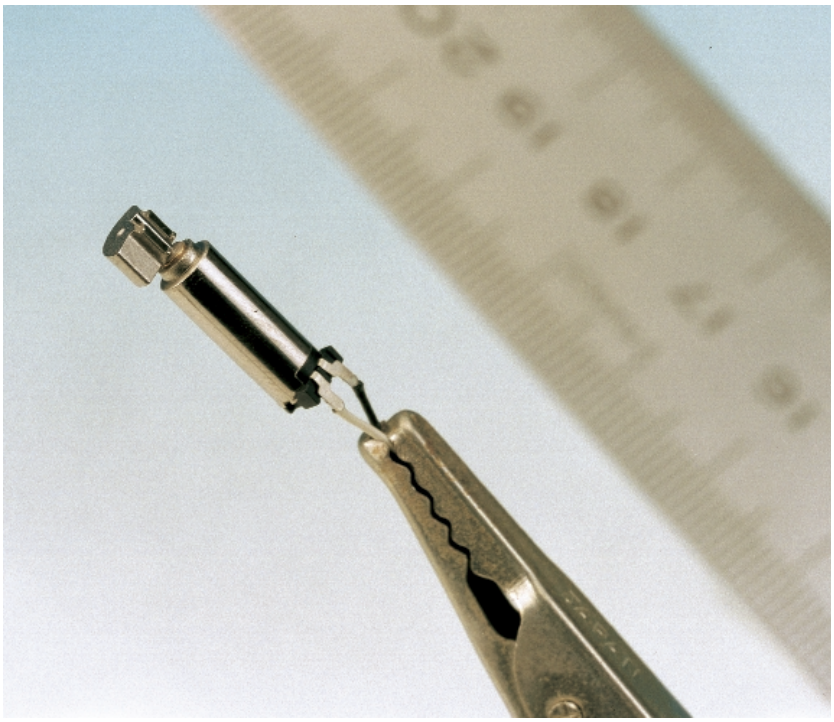
Typ	$G = V_{out} / V_{sense}$
MAX 4173T	+20 V/V
MAX 4173F	+50 V/V
MAX 4173H	+100 V/V

Table 2. Recommended values for R_{sense}.

I_{Lmax}	R_{sense}	$P_{v,max}$	$V_{out,max}$ @ G = 20 V/V (MAX 4173T)	$V_{out,max}$ @ G = 50 V/V (MAX 4173F)	$V_{out,max}$ @ G = 100 V/V (MAX 4173H)
0.1 A	1 W	100 mW	2.0 V	5.0 V	10.0 V
1 A	0.1 W	100 mW	2.0 V	5.0 V	10.0 V
5 A	0.02 W	100 mW	2.0 V	5.0 V	10.0 V
10 A	0.01 W	100 mW	2.0 V	5.0 V	10.0 V

Vibrating batteries for GSM phones

By E. Krempelsauer



It's often the small, everyday things that turn out to have interesting features when you examine them more closely. In any case, here you can learn how vibration alarms and other mobile phone call annunciators work, and how GSM phone batteries are constructed.

Figure 1. The secret of the vibration alarm revealed: a tiny DC motor with an eccentric weight on its shaft.

Maybe you've already wondered how a vibration alarm in a GSM phone actually works. Obviously, it must be small, but what else can you say about it? Want causes it to vibrate? An oscillating coil perhaps, or an oscillating armature? Or is it perhaps a miniature motor with an eccentric weight?

Well, did you guess right? **Figure 1** shows the secret of the vibrating battery: a tiny DC motor with a small weight attached to its shaft so that it is unbalanced when it rotates, and thus vibrates. The example in the photograph was kindly made available to us by Panasonic. The diameter of this mini-motor is only 3.8 mm, and its overall length (including the shaft and eccentric weight) is around 16 mm. The tiny weight is 3 mm long, and it has an outside diameter of 3 mm and an inside diameter of 1 mm. It rotates at approximately 10,000 rpm in order to get your attention. At its rated supply voltage of 1.3 V (working range 0.6 V to 1.6 V), the motor draws at least 65 mA. If you visit the web site of Global Sources (*www.globalsources.com*) and enter 'vibration motors' as the search string, you will see that most Japanese and Chinese manufacturers offer a large assortment of vibration motors of all types.

The photograph in **Figure 2** shows how such a motor (in this case a somewhat larger model) is integrated into a GSM battery package. We simply chiselled open up a standard vibrating battery for a Nokia 6110 GSM phone, and *voilà:* you can't help noticing the vibration motor, which takes up a relatively large amount of space. Even so, there's no overcrowding, since a Sony lithium-ion battery (3.6 V/1000 mAh) has been used here. This is only half as big as the standard set of three NiMH penlight cells normally used with the 6110, which pretty well fill the enclosure of the battery pack. **Figure 3** shows what this type of battery pack looks like when fitted with NiMH cells. The cell block, which consists of five cells for a voltage of 6 V (for a Nokia 3110, for example), is glued into the upper shell of the battery enclosure using a special heat-resistant adhesive. Prior to this, the individual cells are glued to each other and interconnected using welded-on metal

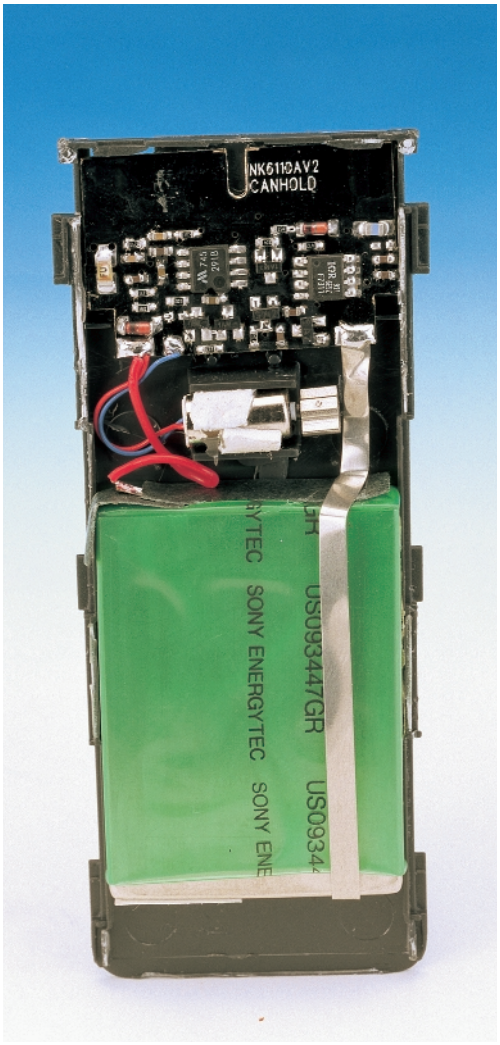


Figure 2. What's inside a standard commercial vibration battery. The vibration motor is mounted between the SMD circuit board containing the charge control circuit and the actual battery, which is a 3.6-V, 1-Ah lithium-ion cell.

strips. A polyswitch is also welded to the assembly to provide short-circuit protection, as well as a thermal cutout that protects against overcharging. In addition, a flexible printed circuit that houses a temperature sensor for monitoring the battery temperature is welded to the cell block. Incidentally, batteries for laptop computers that contain NiMH cells are also constructed in this manner.

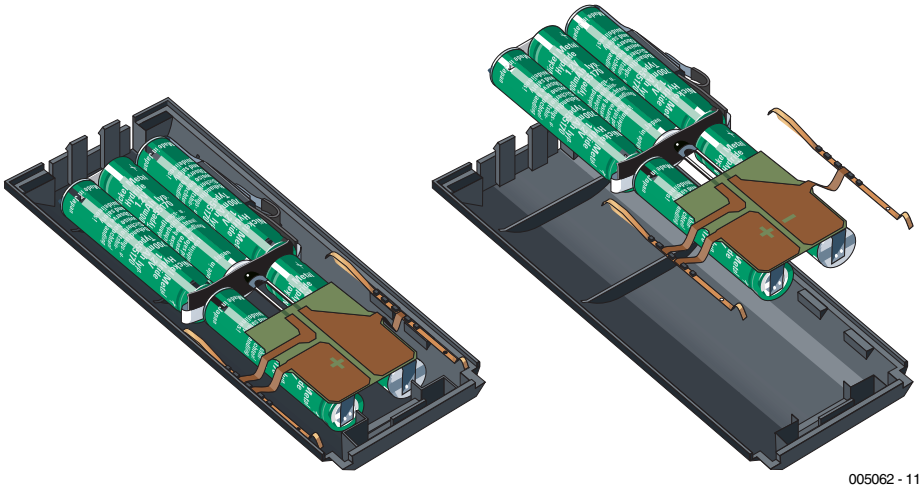
If you compare **Figure 2** with this, you can see that the amount of supplementary circuitry needed for a lithium-ion battery is even greater. Here you can see a densely populated little SMD circuit board that holds the charge control circuit and also provides connections for the vibration motor.

The vibration motor is normally driven by the GSM phone via an electrical contact on the battery pack when the acoustic alarm is switched off.

Clearly, a different solution must be used with separate call annunciators, which can be inexpensively purchased as accessories for GSM phones. These have both an optical indicator (LED) and a vibration alarm. In this case there seems to be a mysterious wireless link between the GSM phone and the annunciator. The mystery is deepened by the fact that simple annunciators in the form of a ballpoint pen with a blinking light are available for a only few pounds. This suggests that the construction of the sensor circuit must be very simple and inexpensive. And indeed, such annunciators contain only an simple broadband field strength detector, consisting of an aerial, a resonant circuit and a Schottky diode, that drives the LED or vibration motor via a simple bit of circuitry whenever the GSM phone transmits a signal. The annunciator gives the impression that it indicates that the phone is ringing. However, this is essentially due to the fact that when an incoming call is detected, the GSM phone diligently sends data bursts to the nearest base station during the 'ringing' interval in order to request a free channel from the network and set up the connection (Channel Request and SDCCH). Since the annunciator only detects that the GSM phone is transmitting, it also reacts when the phone is not ringing, if the phone transmits to the base station for some other reason. This can for example happen during the sign-on sequence after the phone is switched on, when the phone is searching for the network and when the location changes (Location Update). Since these simple annunciators have relative non-selective signal detectors, they can also respond to other transmitters and HF noise signals, such as are radiated in the vicinity of fluorescent lamps.

(005062-1)

Figure 3. Construction of a battery pack with NiMH cells.



005062 - 11

RSC-164

Integrated Circuits Special Functions

ELEKTOR ELECTRONICS

DATASHEET 4/2000

RSC-164
General Purpose Microcontroller Featuring Speech Recognition, Speech & Music Synthesis, Speaker Verification and Audio Record/Playback

Manufacturer
Sensory, Inc., 521 East Weddell Drive, Sunnyvale, CA 94089, USA. Tel. (408) 744-9000, fax (408) 744-1299. Internet: http://www.sensoryinc.com

Application example
Voice recognition with VoiceDirect, Elektor Electronics May 2000.

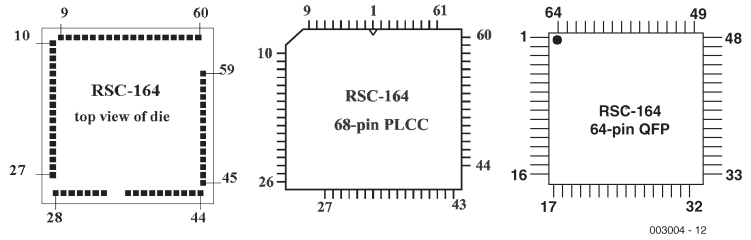
General description
The RSC-164, from the Interactive Speech™ family of products, is a low-cost microcontroller designed for use in consumer electronics. The RSC-164 combines an 8-bit microcontroller with high-quality speaker-independent and speaker-dependent speech recognition, speech synthesis, speaker verification, four-voice music synthesis, and voice record and playback. Products can use one or all of the RSC-164 features in a single application.
The RSC-164 employs a sophisticated neural network that learns to classify sound data. On-chip speech recognition algorithms reach an accuracy of greater than 96% for speaker-independent recognition and greater than 99% for speaker-dependent recognition. Sensory's neural network approach (patent pending) eliminates the need for expensive signal processing or extensive RAM storage.
The highly-integrated nature of the chip reduces external parts count. A complete system may be built with few additional parts other than a battery, speaker, microphone, and audio input support circuitry. Low power requirements make the RSC-164 an ideal solution for battery-powered and hand-held devices.

Features
Full Range of Speech Capabilities
- Speaker-independent speech recognition
- Speaker-dependent speech recognition
- High quality speech synthesis and sound effects
- Speaker verification
- Four-voice music synthesis
- Voice record & playback

Integrated Single-Chip Solution
- 4 MIPS 8-bit microcontroller
- On-chip A/D and D/A converters, digital filtering
- 32kHz clock for time keeping
- Internal 64kbytes ROM; 384 bytes RAM
- 16 general purpose I/O lines
- External memory bus: 16-bit Address, 8-bit Data
- On-chip output amplifier for direct speaker drive

Low Power Requirements
- 3.5 - 5.0V supply
- approx. 10mA operating

RSC-164 Instruction set
The instruction set for the RSC-164 has 52 instructions comprising 8 move, 7 rotate, 11 branch, 11 register arithmetic, 9 immediate arithmetic, and 6 miscellaneous instructions. All instructions are 3



Die-bond pad, PLCC and QFP pin drawings

RSC-164

Integrated Circuits Special Functions

ELEKTOR ELECTRONICS

DATASHEET 4/2000

Pin Descriptions

Name	PLCC Pin/Die Pad	QFP	Pin Description	I/O
AGND	64	52	Analog Ground. For noise reasons, analog and digital grounds should connect together only at the RSC-164.	-
A[15:0]	10-17, 20-27	1-8, 11-18	External Memory Address Bus	O
AIN0	63	51	Analog In, low gain. (range AGND to AVDD/2)	I
AIN1	62	50	Analog In, hi gain (8x input amplitude of AIN0, same range)	I
AVDD	67	55	Analog Voltage. For noise reasons, keep this supply independent of digital circuitry.	-
PWM0	65	53	Pulse Width Modulator Output0	O
DACOUT	60	48	Analog Output (unbuffered).	O
D[7:0]	2-9	57-64	External Data Bus	I/O
GND	1, 18, 33, 52	9, 22, 41, 56	Digital Ground. CPU core (pins 1 and 33) and I/O (pins 18 and 52)	-
PDN	57	NA	Power Down. Active high when powered down.	O
P1[7:0], P0[7:0]	35-42, 43-50	24-31, 32-39	General Purpose Port I/O. Pin P0.0 can act as an external interrupt input. All I/O pins can act as 'wake up' inputs.	I/O
-RDC	53	42	External Code Read Strobe	O
-RDD	55	44	External Data Read Strobe	O
-RESET	32	21	Reset	I
SH	61	49	Sample and Hold. Connect a 470 pF capacitor from here to AGND.	I
-TEI/PWM1	66	54	Test Mode or Pulse Width Modulator Output1 (multiplexed)	I or O
VDD	34, 68	23	Digital Supply Voltage (core)	-
VDDi	19, 51	10, 40	Digital Supply Voltage (I/O line)	-
-WRC	54	43	External Code Write Strobe	O
-WRD	56	45	External Data Write Strobe	O
-XMH	58	46	External Hi-memory enable (low active)	I
-XML	59	47	External Low-memory enable (low active)	I
XO1	30	19	Oscillator 1 output (high frequency)	O
XI1	31	20	Oscillator 1 input	I
XO2	29	NA	Oscillator 2 output (32768 Hz)	O
XI2	28	NA	Oscillator 2 input	I

bytes or fewer, and no instruction requires more than 8 clock cycles to execute.

General purpose I/O
The RSC-164 has 16 general purpose I/O pins (P0.0-P0.7, P1.0-P1.7). Each pin can be programmed as an input with weak pull-up (~200kΩ equivalent device); input with strong pull-up (~10kΩ equivalent device); input without pull-up, or as an output. This is accomplished by having 32 bits of configuration registers for the I/O pins (Port Control Register A and Port Control Register B for ports 0 and 1).

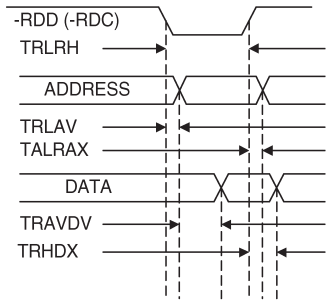
External memory
The RSC-164 includes an external memory interface that allows connection with memory devices for speaker-dependent speech recognition, audio record/playback, extended durations of speech and music synthesis, and enhanced product functionality. Separate data and address buses allow use of standard EPROMs, ROMs, SRAMs, and flash memory with little or no additional decoding. Provision of separate read and write signals for each external memory space further simplifies interfacing. The RSC-164 includes 8 data lines (D[7:0]) and 16 address lines (A[15:0]), along with associated control signals for interfacing to external memory. Using flash memory and EEPROM will require custom code development. The RSC-164 can connect serially through two I/O lines to a serial EEPROM for applications with low data storage requirements.

Oscillators
Two independent oscillators in the RSC-164 provide a high-frequency clock and a 32kHz timekeeping clock. The oscillator characteristics are as follows:
- Oscillator #1: Pins XI1, XO1
  14.32 MHz (3.5V-5.0V)
- Oscillator #2 Pins XI2 and XO2
  32768 Hz (3.5V-5.0V)

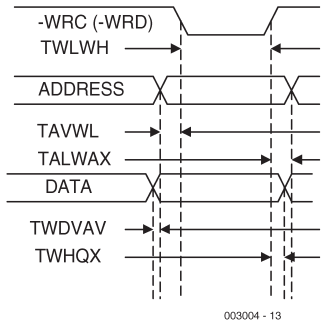
Oscillator #1 works with an external crystal, a ceramic resonator or LC. Use of Oscillator #2 requires a crystal for precision timing.

Clock
The RSC-164 uses a fully static core — the processor can be stopped (by removing the clock source) and restarted without causing a reset or losing contents of internal registers. Static operation is guaranteed from DC to 14.32 MHz.
Typically the processor clock runs from a 14.32 MHz crystal with no divisor and one wait state. This creates internal RAM cycles of 70 nsec duration and internal ROM or external cycles of 140 ns duration. Careful design of external decoding logic and close analysis of gate delays may allow operation with memories having access times as slow as 120 ns.

Timers/counters
The two independent oscillators of the RSC-164 provide counts to two internal timers. Each of the



External read timing



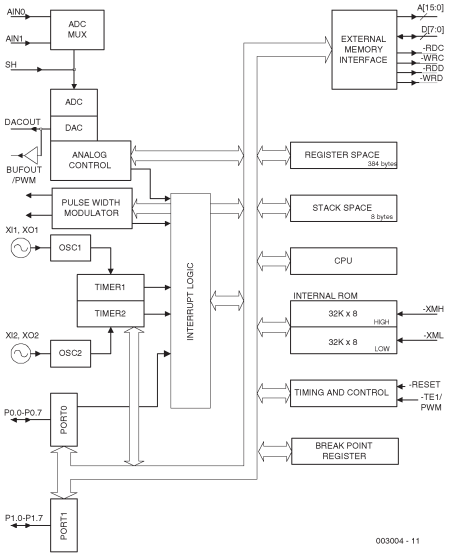
External write timing

RSC-164
Integrated Circuits
Special Functions
ELEKTOR ELECTRONICS
DATASHEET 4/2000

two timers consists of an 8-bit reload value register and an 8-bit up-counter.
The reload register is readable and writeable by the processor.

Interrupts
The RSC-164 allows for five interrupt sources, as selected by software. Each has its own mask bit and request bit in the IMR and IRQ registers respectively.
The following events can generate interrupts:
- Positive edge on Port 0, bit 0
- Overflow of Timer 1
- Overflow of Timer 2
- Sensory reserved functions
- Completion of PWM sample period

Analog output
The RSC-164 offers two separate options for analog output. The DAC (Digital to Analog Converter) output provides a general purpose 10-bit analog output that may be used for speech output (with the inclusion of an audio amplifier), or other purposes requiring an analog waveform. For speech applications that require driving a small speaker, the PWM (Pulse-Width Modulator) output can be used instead of the DAC output. The PWM output can directly drive a 32 ohm speaker.



RSC-164 Architecture

DC Characteristics (T0 = 0°C to +70°C, Vdd = 5V)

SYMBOL	*PARAMETER*	*MIN*	*TYP*	*MAX*	*UNITS*	*TEST CONDITIONS*
V_{IL}	Input Low Voltage	–0.1		0.75	V	
V_{IH}	Input High Voltage	2.5		Vdd+0.5	V	
V_{OL}	Output Low Voltage		0.3	0.5	V	I_{OL} = 4 mA
V_{OH}	Output High Voltage	4.0	4.3		V	I_{OL} = –4 mA
I_{IL}	Logical 0 Input Current					
I_{CC1}	Digital Supply Current		10		mA	Osc1 Freq=14.32 MHz, CPU clock divide by 1
I_{CC2}	Analog Supply Current		0.15		mA	Osc1 Freq=14.32 MHz, CPU clock divide by 1
I_{CC3}	Digital Supply Current, Standby					Power-down mode
I_{CC4}	Analog Supply Current, Standby					Power-down mode
Rpu	Pull-up resistance P0.0-P1.7	10	400	Hi-Z	kΩ	selected with software

part 1: introduction and operation

hotkeys keyboard

with 18 freely programmable keys

Design by B. Stuurman

What's this, a professional-looking keyboard with no legends on the keycaps? This is intentional, so that you can choose the function performed by each key. It's even possible to assign a complete set of keystrokes to a single key. In part 1 of this article, we discuss the communications between a normal keyboard and a PC, describe how a hotkeys keyboard can very cleverly worm its way into this process, and conclude with a complete description of how to use the hotkeys keyboard. In part 2 next month, we will look at the technical details.

One of the annoying features of many Windows programs is that a whole series of keys must be pressed to invoke certain commands. Consider PageMaker, for example: to use the magnifying glass, which happens all the time, you have to first press the Shift key, then the Alt key and finally the F7 key. This means that you have to hold all three keys down at the same time. Think also of the special characters, for which you have to first press the Alt key and then enter a three-digit or five-digit number using the numeric keypad.

All of this is no longer necessary with the hotkeys keyboard. You can program it for each application with the key combinations that you use most often, upload this information to the hard disk, and download it again whenever you want to use it.

Templates

Since the functions of the keys of the hotkeys keyboard depend on the application with which they are used, it does not make any sense to have any legends on the keycaps. The layout of the keyboard is designed such that it is very easy to

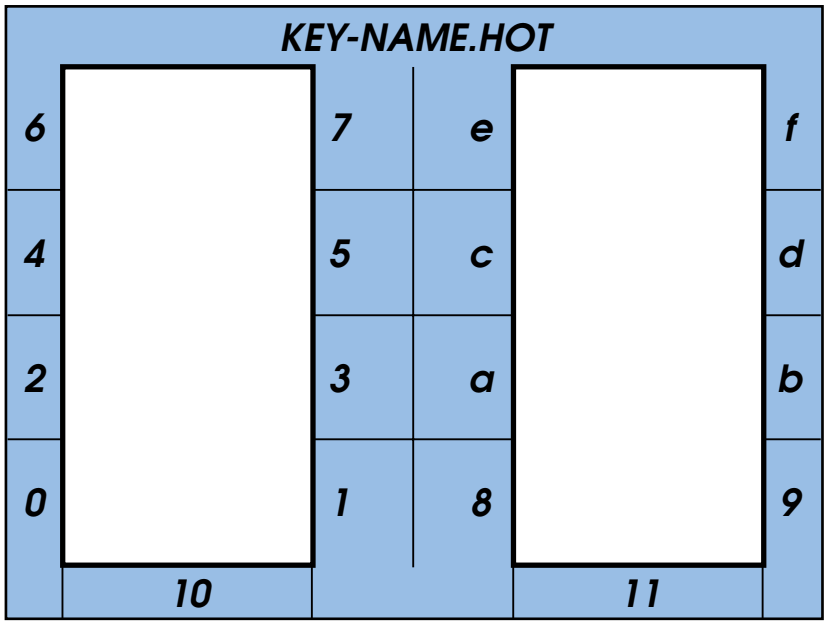


Figure 1. A template with the 'name' of each key is necessary so that you can program the keys.

use templates with it. The function of each key can be marked on the template next to the key. The name of the associated hotkeys file stored on the hard disk can also be written on the template. This makes it very easy to load the proper hotkeys file for a particular application.

Even though the hotkeys do not have any fixed functions, they still have to have names so that they can be programmed. We have chosen to name them using the hexadecimal numbers 0 – 9, a – f, 10 and 11. This amounts to eighteen numbers. The hotkeys keyboard actually has 16 keys, but two more 'keys' can be formed by simultaneously pressing the bottom-most keys of the left-hand part (10) or the right-hand part (11). This yields a total of 18 keys.

When the hotkeys keyboard is switched on for the very first time, the names and functions are assigned to the keys. This is because the program 'sees' that the EEPROM in which the key strings are stored is empty, and so it copies a table from the ROM to the EEPROM. Even at this stage, a template is necessary to mark the names of the keys, so that you can identify the key that you want to program. Figure 1 shows this template. You can use a photocopier to enlarge it to the specified size, and then glue it to a thin piece of cardboard. After this, you have to cut out the openings for the keys. Full-sized versions of the template in several different file formats are also located on the diskette for this project. You can use this template as a model for making your own templates. A graphics program such as Windows Draw or CoreDRAW is very suitable for this purpose.

Communications between a PC keyboard and a PC

The communications between a standard PC keyboard and a PC are more complicated than you might initially assume. This is because the keyboard not only has to send data to the PC, but the PC can also send data to the keyboard. Let's consider an example.

If the Caps Lock key on the keyboard is pressed, the keyboard sends the 'make' code for this key (58_H) to the PC. The PC responds to this by setting the associated bits in the keyboard status register. After this, the PC sends the keyboard a command to switch on the Caps Lock LED (ED_H, 04_H). The keyboard responds with an Ack(knowledge) character (FA_H) after each byte. Once all the data are received, the keyboard switches on the Cap Locks LED. Try the following experiment: press the Caps Lock key, and note that the Caps Lock LED is illuminated. Now unplug the keyboard connector from the PC, wait a little while and plug it in again. The Caps Lock LED on the keyboard will be off, since the keyboard has been reset. Now press a key, such as 'm', and what do you see on the monitor? That's right, it's an 'M', because the Caps Lock status bit in the PC has not been changed.

Each key on the PC keyboard has its own individual code. When the key is pressed, the keyboard sends this code, which is called the 'make code', to the PC. When the key is released, the same code is again sent to the PC, but this time the 'break code prefix' ($F0_H$) is added in front of the code.

For example, the make code of the left Shift key is 11_H. If this key is pressed, the keyboard sends 11_H to the PC. If the Shift key is held pressed, the keyboard sends a whole series of 11_H codes to the PC, since the repeat code is the same as the make code. When the Shift key is released, the keyboard sends $F0_H$, 11_H to the PC. This is a very important piece of information. The only thing that the keyboard sends to the PC is a make code when a key is pressed and a break code when a key is released.

Now we know how to make an upper case letter. All we have to do is to first send the make code of the Shift key, followed by the make and break codes of the letter in question, and finally the break code of the Shift key. If we know that the make code of the 'm' key is $3A_H$, we could program hotkey 0 with following string of codes:

$$11_H,\ 3A_H,\ F0_H,\ 3A_H,\ F0_H,\ 11_H$$

And indeed, if we now press key 0 of the hotkeys keyboard, we will see an 'M' appear on the screen.

All this means that we need to know the make codes of the keys on the PC keyboard in order to be able to program the hotkeys keyboard. The diskette for this project includes a table that lists all the relevant keys and their associated make codes (see also the summary in the article 'PC keyboard encoding' in the February 2000 issue of *Elektor Electronics*). However, the hotkeys keyboard can also tell you itself which code belongs to a particular key, as we will explain shortly.

You may wonder why we referred to the 'relevant keys'. The original keyboards that were delivered with PCs had a block of ten function keys to the left of the alphanumeric keys. These keyboards had 84 keys. Later on, there came keyboards with the function keys in a row across the top and some additional keys, such as the cursor keys. These are called 'extended' keyboards. The make codes for the 'extended' keys are different from the 'basic' make codes, in that they are preceded by an additional code, such as $E1_H$ or $E0_H$. This is not all that important, but it is nice to know that the hotkeys keyboard can also deal with this.

Sending codes to the PC

We now know how to make the strings that we want to send to the PC, but we don't yet know how this is supposed to happen. The PC keyboard

Command summary

keybeep?
This is a switch (alternate-action) command that controls whether or not the hotkeys keyboard emits a beep whenever a hotkey is pressed.

repeat?
This is a switch command that enables or disables the hotkey repeat function.

reseep?
This command resets the hotkeys to their initial states. In other words, the default table is copied from the ROM to the EEPROM. In the initial state, the function of each key is its name.

scankb?
The hotkeys keyboard decouples the PC keyboard from the PC and then listens to the PC keyboard. If a key is pressed on the PC keyboard, the hotkeys keyboard displays the string that it has received on the monitor after the key has been released (see **Figure 4**). The' scankb' mode can only be exited by pressing hotkey 0! (Note: since a certain minimum number of instructions must be executed to receive each bit, the clock period of the PC keyboard may not be less than 80 μs.)

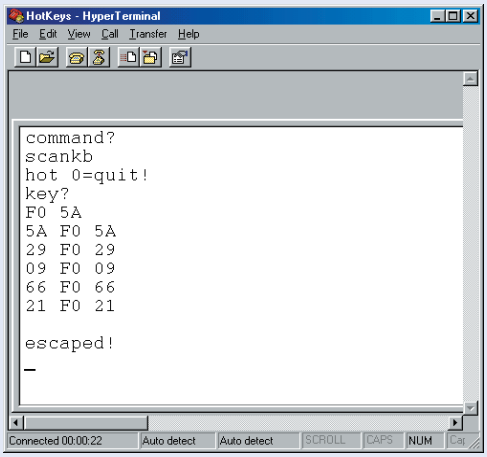


Figure 4. The 'scankb' command in action. The first two codes, 'F0 5A', are caused by releasing the Enter key. Following this, the following keys were pressed and released in turn: Enter, Space, F10, Backspace and Control. You can 'escape' from this command using hotkey 0.

volume?
This command allows the key beeps to be set loud or soft (if they have been enabled).

rtest?
This is a utility command to check whether values of the divider resistors of the hotkeys keyboard are within the allowed tolerances (this is discussed in part 2).

key?
This command has two options. If only the 'name' of a hotkey is supplied as a parameter, the make and break codes assigned to that key are displayed (see **Figure 5**). If the 'name' of the hotkey is followed by one or more hexadecimal bytes (that is, with two hexadecimal characters per byte) separated by spaces, these bytes are accepted as the new string to be assigned to that hotkey. It is not necessary (nor is it even possible) to enter the break codes, since these are generated by the hotkeys keyboard itself in the reverse order of the make codes. There are 128 bytes of EEPROM available to store the strings, and to make efficient use of this memory only the make codes are stored. The code '00H' is stored between pair of strings, which means that it is prohibited to enter this code as part of a string. The strings are assigned dynamically; there is not a fixed number of bytes reserved for each hotkey. If the total size of the strings (plus the separator characters) exceeds the amount of available memory, this is reported and the string is not stored.

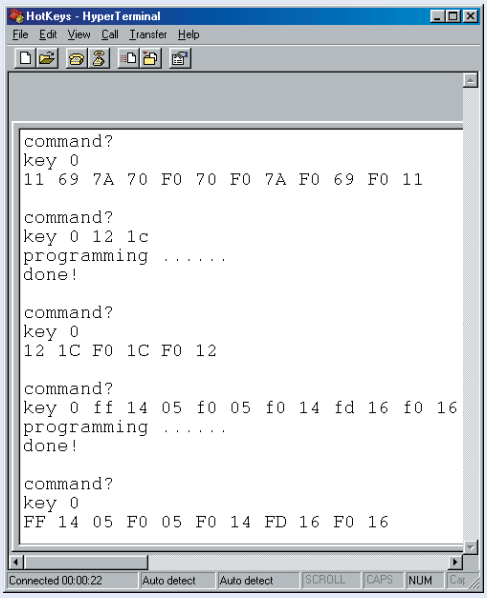


Figure 5. The 'key' command in action. Hotkey 0 is first pressed to display the string assigned to it. Hotkey 0 is then programmed with a different string, which is displayed on the following line. Finally, hotkey 0 is programmed with the 'free format' string in the text. The 'fd' character provides a delay of 250 ms.

The 'key' command also allows a hotkey string to be 'literally' assigned to a hotkey. In this case the string must begin with the code 'ff_H', which can be easily remembered as 'free format'. Such a string must contain the break codes as well as the make codes. Everything is written literally to the EEPROM. When the hotkey in question is pressed, the entire string is output. It is even possible to incorporate a delay in a free-format string. The 'fd_H' code (which you can remember as 'function delay') provides a delay of 250 ms. For example, suppose that a particular program allows you to open a gateway to DOS by holding down Control and pressing the F1 key. This causes the option '*1?to DOS: 0*' to appear. This means that the '1' key must be pressed to go to DOS. This whole scenario can be assigned to a single hotkey in the form of a free-format string. Since you know (by now) that 'Contr'=14_H, 'F1'=05_H en '1'=16_H, you can program hotkey 0 for this purpose as follows:

```
key 0 ff 14 05 f0 05 f0 14 fd 16 f0 16
```

The 'fd_H' code thus provides a delay of 250 ms before the code for the '1' key is output.

^d?
The program for the hotkeys keyboard includes Xmodem protocols (with CRC) for sending and receiving hotkey files. If you want to save a hotkey file on the hard disk, you can select the Transfer

menu in HyperTerminal and then the Receive File option. After this you must select a folder and select Xmodem as the protocol. You will then be asked to give a name to the received file. After you click on OK, the file will be copied to the hard disk (including the key beep, repeat and volume settings). In order to make it easier to recognise hotkey files, they are assigned the suffix 'HOT'.
The Xmodem protocol can however only receive data. In other words, if you want to download a hotkey file from the hard disk to the hotkeys keyboard, the initiative must come from the hotkeys keyboard (which will receive the file). Since the hotkeys keyboard normally wants to upload a file to the hard disk, it is necessary to first 'prime' it to receive a file before activating the 'Send' option in HyperTerminal. This is the function of the '^d' (Ctrl d) command. First press this key combination, then select Transfer in the HyperTerminal menu bar and select Send File. Then select the name of the file to be sent and press the Send button. To initiate the actual transfer, you must now press hotkey 1.

??
This causes a command summary to be displayed on the monitor.

sends its data to the PC in serial form. The protocol is synchronous, which means that the data bits are transferred under the control of a clock signal. This means that the data transmission timing is not critical. In practice, you can find keyboards with clock periods ranging from 70 µs to 120 µs. Figure 2 shows the serial data transfer format. In the rest state, the data and clock lines are both high, since the PC and the keyboard both have open-collector outputs and there are pull-up resistors at both ends. The transmission begins with a start bit ('0'), which is followed by the data bits D0 through D7, then a parity bit (uneven) and finally a stop bit ('1'). The data must be stable on the negative edge of the clock. From Figure 2, you can also see that the changes in the level of the data signal occur when the clock signal is high. The make code of the Return key ($5A_H$) is shown for illustration in this diagram.

The hotkeys microcontroller has send and receive routines for this data transfer format. After it receives a byte, the PC needs a certain amount of time to process it, and the keyboard is not allowed to send any data during this time. The PC indicates this to the keyboard by pulling the clock line low. This occurs during the eleventh clock interval. Once the PC is again free to receive data, it lets the clock line go high. Naturally, the same thing applies in reverse to the hotkeys keyboard. After it has sent a byte, it must wait until the PC has released the clock line before sending the next byte. On the other hand, if the hotkeys keyboard receives data from the PC keyboard, it can hold the clock line to the PC keyboard low so that it has time to put the scan codes in its buffer. Once it is finished with this, it can allow the clock line to go high again.
Just to avoid any confusion, we should note that when the PC sends data to the keyboard, the clock signal is supplied by the keyboard. This is thus a different transmission format!

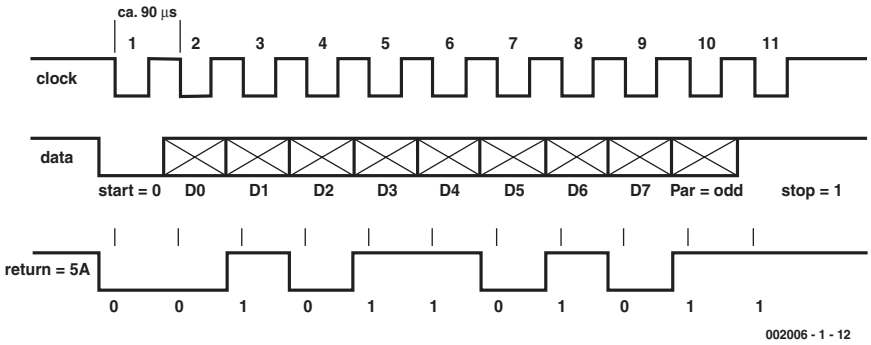


Figure 2. The serial data transfer format that the PC keyboard uses to send information to the PC.

Operating instructions

In *The Advanced Programmer's Guide* from Ashton-Tate, it is suggested that you should first write the operating instructions before you start programming. This sounds a bit strange, since how can you write the operating instructions for something that does not yet exist! Nevertheless, it is a very good idea, since it forces you to carefully think about the program you want to produce before you start writing it. In fact, you have to think about the program so much that once you have drawn up the operating instructions, you have just about mapped out the actual program.

We can now carry on with the operating instructions for the hotkeys keyboard, and leave the description of the hardware to next month.

Connecting the hotkeys keyboard to your system is very simple. Plug the cable with the male DIN connector in the keyboard socket of the PC, and insert the plug from the PC keyboard in the matching connector of the hotkeys keyboard. The hotkeys keyboard thus sits between the PC keyboard and the PC. In addition, you will need a serial extension cable to connect the hotkeys keyboard to a serial port of the PC. A 9-pin female D-type connector is provided for this at the back of the circuit board.

All signals from the PC keyboard are simply passed through the hotkeys keyboard to the PC, with the understanding that the clock and data signals pass through low-impedance analogue switches located on the controller circuit board of the hotkeys keyboard. The hotkeys keyboard also taps its power from the supply lines for the PC keyboard.

Under normal circumstances, the hotkeys keyboard does not interfere with the communications between the PC keyboard and the PC. It only takes action in two situations:

1. If a hotkey is pressed, the hotkeys keyboard breaks the connection between the PC keyboard and the PC and sends the make codes of its key to the PC. When the hotkey is released, the break codes are sent. The circuit is however smart enough to monitor the clock line, so it knows when it has to wait. Finally, it makes itself invisible again by closing the analogue switches.
2. The hotkeys keyboard also breaks the connection between the PC keyboard and the PC when a 'scankb' command is being executed. In this case, however, it waits for a key on the PC keyboard to be pressed. When the key is released, the hotkeys keyboard displays the received scan code string on the screen.

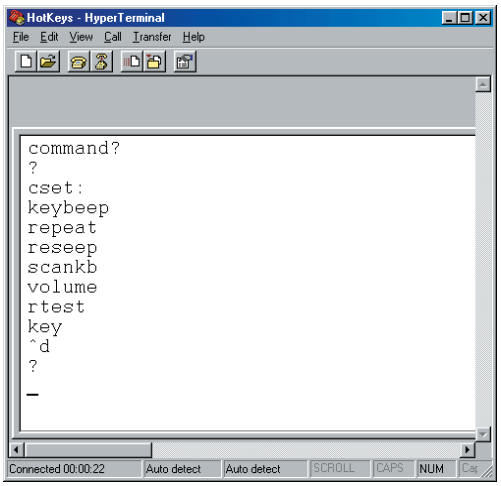


Figure 3. The command set of the hotkeys keyboard.

A serial communications link is necessary between the hotkeys keyboard and a COM port of the PC in order for the PC to communicate with the hotkeys keyboard. In addition, there must be a terminal emulator program present in the PC, such as the HyperTerminal program belonging to Windows. With a standard Windows installation, you will find HyperTerminal under Start → Programs → Accessories → HyperTerminal.

Configure the selected COM port for 19200, 8, n, 2. It is usually necessary to save these settings and restart HyperTerminal in order to make the new settings effective. An older terminal emulator program such as Procomm can also be used, but Windows 98 does not work very well with DOS programs (Procomm 'sees' the serial port only after the diagnostic data for the COM port in question has been requested in the 'Modems' configuration panel).

Now that everything is installed and the terminal emulator program is running, it is possible to communicate with the hotkeys keyboard. Press the Enter key, and the hotkeys keyboard will answer with 'command?'. If you want to know what commands are supported by the hotkeys keyboard, press '?' followed by Enter. The hotkeys keyboard responds by displaying its complete command set on the screen (see Figure 3).

As you can see, the hotkeys keyboard employs only lower-case letters, and the same thing applies to the user. Upper-case letters are forbidden, even for hexadecimal numbers!

The commands are all explained in the accompanying text box. Here we can make two comments:

1. The hotkeys keyboard contains an interactive command processor. Once you have typed a command, including any parameters, and finished by pressing Enter, the hotkeys keyboard investigates the entire command line and returns a clear error message if necessary. If there is no error in the command line, the command is executed. Note that the Backspace and Delete keys have no effect; if you have to 'bail out' of the program you can press the Esc key.

2. The hotkeys keyboard will work properly even without a serial connection to a COM port. However, in this case it is not possible to communicate with the hotkeys keyboard, and it is not possible to change the key string table 'on the fly'.

Now you know nearly everything about what the hotkeys keyboard can do. All that's missing is the hardware, which we will examine next month in part 2.

(002006-1)

CONSTRUCTION GUIDELINES

Elektor Electronics (Publishing) does not provide **parts and components other than** PCBs, fornt panel foils and software on diskette or IC (not necessarily for all projects). Components are usually available form a number of retailers – see the adverts in the magazine.

Large and small values of components are indicated by means of one of the following prefixes :

E (exa) = 10^{18}	a (atto) = 10^{-18}
P (peta) = 10^{15}	f (femto) = 10^{-15}
T (tera) = 10^{12}	p (pico) = 10^{-12}
G (giga) = 10^{9}	n (nano) = 10^{-9}
M (mega) = 10^{6}	μ (micro) = 10^{-6}
k (kilo) = 10^{3}	m (milli) = 10^{-3}
h (hecto) = 10^{2}	c (centi) = 10^{-2}
da (deca) = 10^{1}	d (deci) = 10^{-1}

In some circuit diagrams, to avoid confusion, but contrary to IEC and BS recommandations, the value of components is given by substituting the relevant prefix for the decimal point. For example,

3k9 = 3.9 kΩ 4μ7 = 4.7 μF

Unless otherwise indicated, the tolerance of resistors is ±5% and their rating is ⅓–½ watt. The working voltage of capacitors is ≥ 50 V.

In **populating a PCB**, always start with the smallest passive components, that is, wire bridges, resistors and small capacitors; and then IC sockets, relays, electrolytic and other large capacitors, and connectors. Vulnerable semiconductors and ICs should be done last.

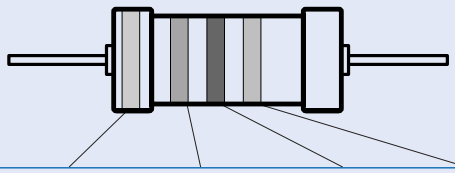
Soldering. Use a 15–30 W soldering iron with a fine tip and tin with a resin core (60/40) Insert the terminals of components in the board, bend them slightly, cut them short, and solder: wait 1–2 seconds for the tin to flow smoothly and remove the iron. Do not overheat, particularly when soldering ICs and semiconductors. Unsoldering is best done with a suction iron or special unsoldering braid.

Faultfinding. If the circuit does not work, carefully compare the populated board with the published component layout and parts list. Are all the components in the correct position? Has correct polarity been observed? Have the powerlines been reversed? Are all solder joints sound? Have any wire bridges been forgotten?

If voltage levels have been given on the circuit diagram, do those measured on the board match them – note that deviations up to ±10% from the specified values are acceptable.

Possible corrections to published projects are published from time to time in this magazine. Also, the readers letters column often contains useful comments/additions to the published projects.

The value of a resistor is indicated by a **colour code** as follows.



color	1st digit	2nd digit	mult. factor	tolerance
black	–	0	–	–
brown	1	1	×10^{1}	±1%
red	2	2	×10^{2}	±2%
orange	3	3	×10^{3}	–
yellow	4	4	×10^{4}	–
green	5	5	×10^{5}	±0,5%
blue	6	6	×10^{6}	–
violet	7	7	–	–
grey	8	8	–	–
white	9	9	–	–
gold	–	–	×10^{-1}	±5%
silver	–	–	×10^{-2}	±10%
none	–	–	–	±20%

Examples:
brown-red-brown-gold = 120 Ω, 5%
yellow-violet-orange-gold = 47 kΩ, 5%