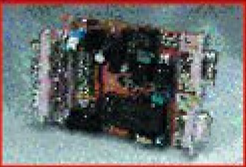
ELEKTOR ELECTRONICS

THE ELECTRONICS & COMPUTER MAGAZINE

JUNE 2000
£2.95

www.elektor-electronics.co.uk

Magnetic Card Reader



PC Interface for CAN Bus

- WAP Phones
- Intelligent Door Lock
- DVD – the Megastore
- One-Wire Spy

Valve Preamplifier

retro electronics – excellent sound

06
9 770268 451081

Intelligent Door Lock

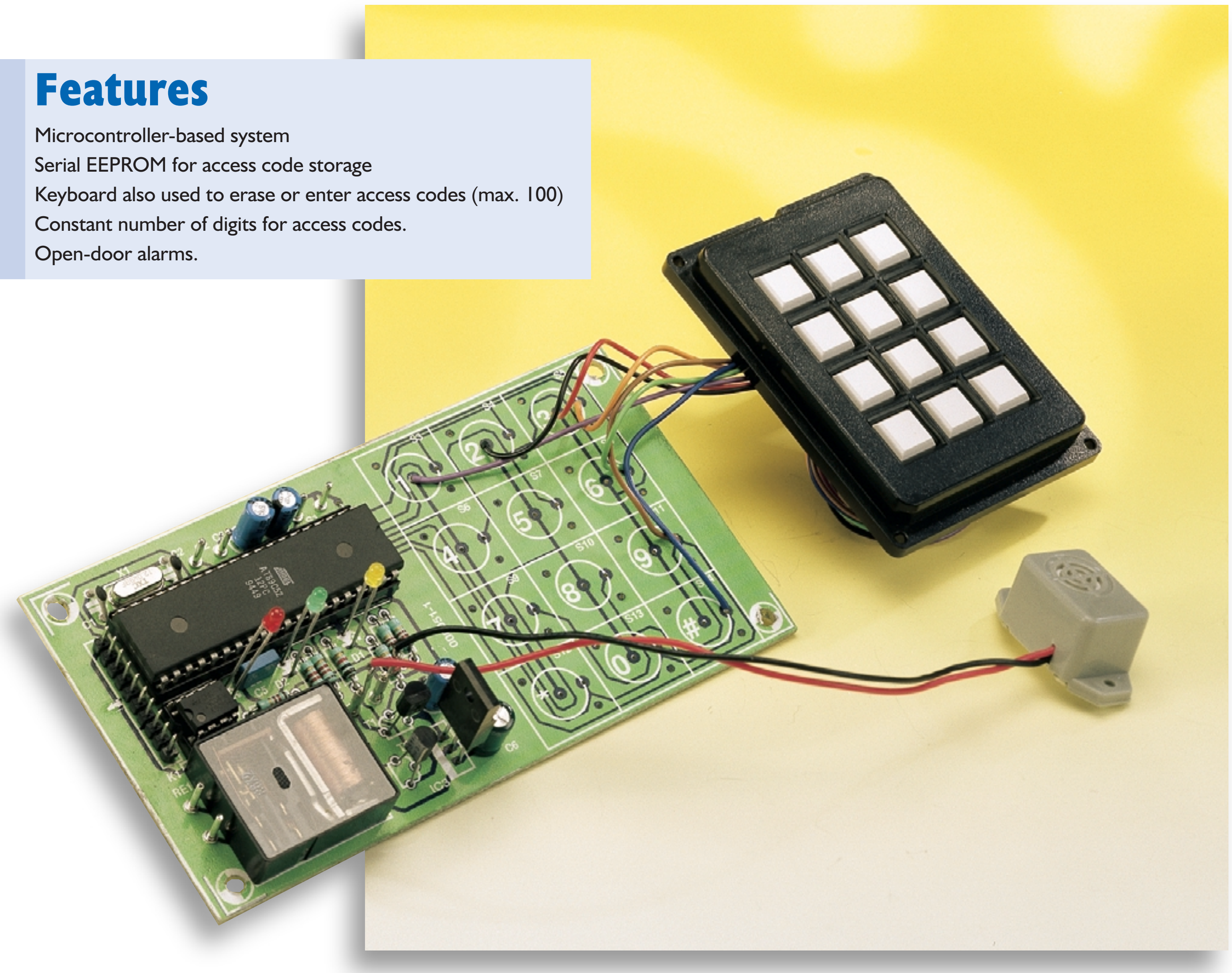
an advanced access control system

By Prof. A. Roldán Aranda – Email: aroldan@uhu.es

The Intelligent Door Lock employs an entrance code to control access to certain rooms or buildings. The heart of the system is a microcontroller manufactured by Atmel. The system employs an electric door latch and has visual and acoustic indicators for ease of use and your personal safety. Access codes are stored in a non-volatile memory device.

Features

Microcontroller-based system
Serial EEPROM for access code storage
Keyboard also used to erase or enter access codes (max. 100)
Constant number of digits for access codes.
Open-door alarms.



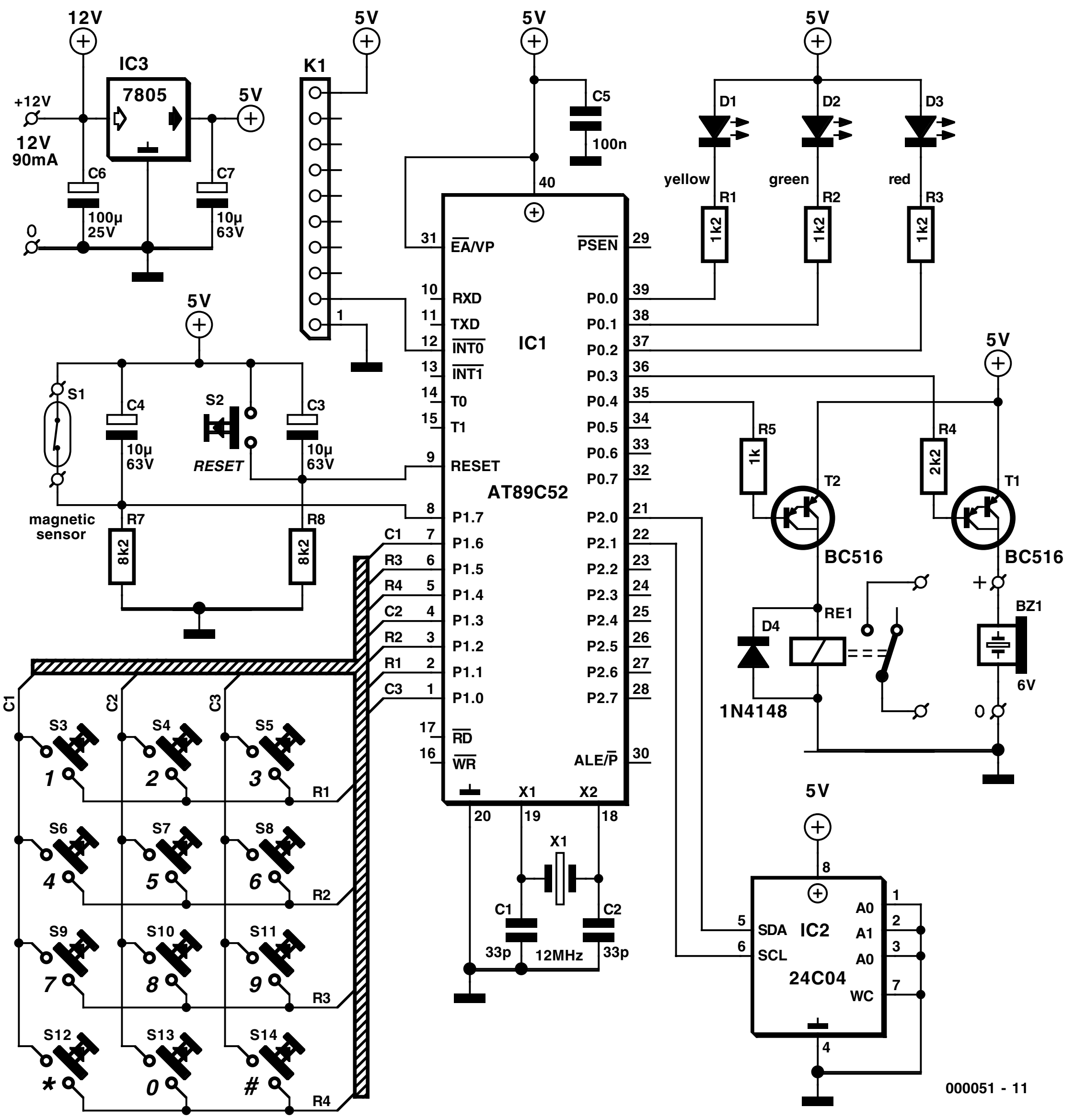


Figure 1. Circuit diagram of the Code Lock.

The flexibility of the system is due to a microcontroller running a program written specifically for the purpose of controlling access to rooms, apartments, hallways and complete buildings.. An all-hardware version of such a system would be much complex and costly. The use of an EEPROM allows access codes to be stored without having to worry about loss of data due to power supply failures.

The access codes can easily be reprogrammed by switching the system into 'supervisor mode', as will be discussed further on.

A magnetic sensor (reed switch) is included to detect if the door is closed or open. The system enters a steady state until the door is closed, and this situation is indicated by a green light. If the door is kept open for a specified time, a buzzer will sound until the door is finally closed.

This alarm can be turned off by connecting the leads of the magnetic sensor on the door.

Basic elements

The different blocks that constitute the system are

1. keyboard and indicators
2. main circuit
3. solenoid-driven door latch
4. magnetic detector (reed switch)

Items 1 and 2 are found back in the circuit diagram shown in **Figure 1**. Microcontroller IC1, a Type AT89C52 from Atmel is the 'decision maker' in this circuit, handling such tasks as scanning the keypad (S3-S14), and reading the status of magnetic sensor S1 (a reed switch) and reset switch S2. It also controls a number of output devices including three status indicator LEDs (D1, D2 and D3), a relay (solenoid) driver (T2), and a buzzer driver (T1) for acoustic signals. System data including the code to open the door are stored in an EEPROM, IC2, which is linked to the microcontroller by two port lines, P2.0 (serial data) and P2.1 (serial clock).

The microcontroller runs at a speed of 12 MHz which is determined by quartz crystal X1 and capacitors C1 and C2.

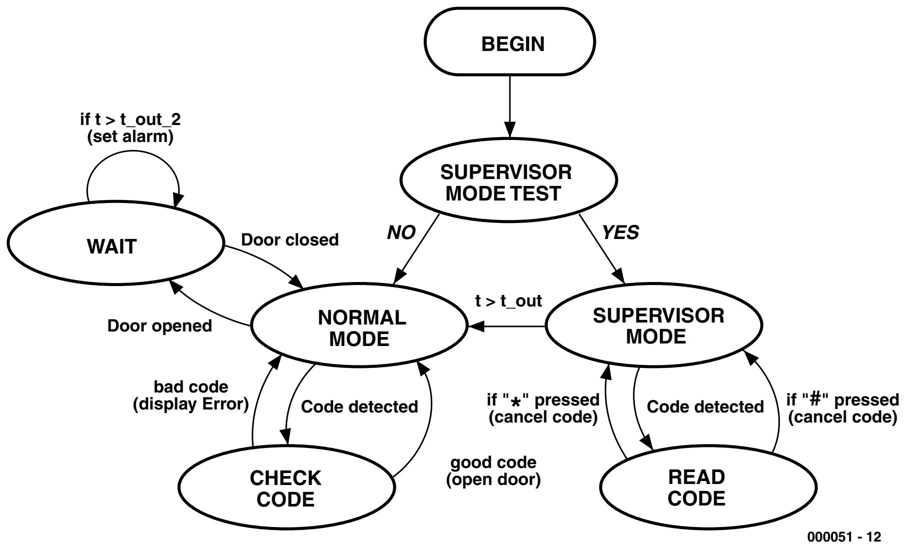


Figure 2. Flow chart of the software running inside the AT89C52 microcontroller.

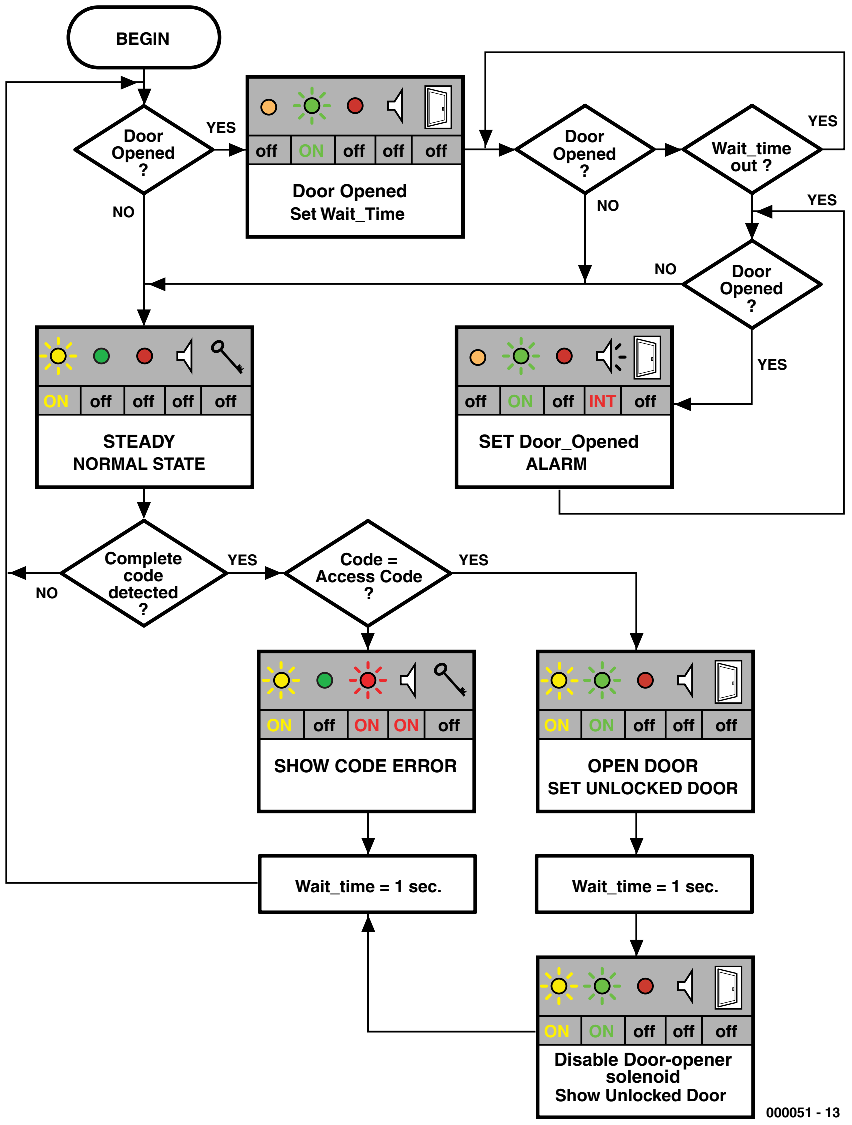


Figure 3. Flowchart and indicator activity during Normal Mode.

The power-on reset circuit is conventional, consisting of electrolytic capacitor C3 and resistor R8. When power is applied to the circuit, the microcontroller's reset input will be held high by C3 until the capacitor is sufficiently charged. The delay introduced in this way ensures that the microcontroller is not enabled (i.e., released from its reset state) until the supply voltage is stable. Pressing S2 allows the circuit to be reset manually.

The circuit has an on-board voltage stabilizer, IC3, which provides a clean 5-V rail for the other parts to work on. Just about any small mains adaptor rated at 9-12V d.c and 150 mA or so should be suitable to power the code lock. Most of the current will, of course, go into actuating the on-board relay, Re1. The electric door latch will typically require its own power supply. Usually this will be 12 V or 24 V, but mains-powered types may also be used if electrical safety precautions are observed and the coil current does not exceed the contact specification of the on-board relay. For safety's sake, however, we'd suggest going for a low-voltage dc-operated door latch.

Software in charge

In this circuit, a microcontroller is used instead of a handful of digital ICs and other hardware to keep the component count low while maximizing the number of ways in which the design can be matched to real-life requirements. In other words, during the design stages, certain new features are far easier to implement by adapting the software than by, say, adding half a dozen components (and changing the board layout, etc.)

A flowchart illustrating the operation of the software packed in the Atmel microcontroller (in the form of an executable program) is shown in **Figure 2**. Some of the labels and descriptions given in this diagram will be discussed below as part of the Mode descriptions.

The system operation consists of two modes, 'normal' and 'supervisor'. **Supervisor mode** is used to update the entrance codes. The only way to enter this mode is by keeping the *, 5 and # keys pressed when starting-up the system. In all other cases, the system enters normal mode. The system will remain in the latter mode unless the triple key combination *5# is detected. The main function of the normal mode is to switch on the solenoid-driven latch unlocking the door (only when the access code is activated) or to inform non-authorized visitors that the code is incorrect. When the door

COMPONENTS LIST

Resistors:
R1,R2,R3 = 1kΩ2
R4 = 2kΩ2
R5 = 1kΩ
R7,R8 = 8kΩ2

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

Semiconductors:
D1 = LED yellow, high efficiency
D2 = LED, green, high efficiency
D3 = LED, red, high efficiency
D4 = 1N4148
T1,T2 = BC516
IC1 = AT89C52-12PC, programmed, order code **000051-41**
IC2 = 24C04
IC3 = 7805

Miscellaneous:
K1 = 10-way SIL header
S1 = Reed switch and permanent magnet for door mounting
S2 = pushbutton, 1 make contact
S3-S14 = pushbutton, 1 make contact, Marquardt type 6425
Basic switch: Conrad Electronics order #70 68 92
Keycaps: Conrad Electronics

# 70 69 06	#70 69 14
#70 69 22	#70 69 30
#70 69 49	#70 69 57
#70 69 65	#70 69 73
#70 69 81	#70 69 90
#70 70 07	#70 70 15

X1 = quartz crystal 12MHz
Bz1 = buzzer, passive type, 6V operating voltage
RE1 = relay, horizontal mounting, 1 changeover contact, coil voltage 6V or 5V, e.g., Siemens V23057-A0001-A101
PCB, order code **000051-1**
Disk (AT89C52 source code), order code **000051-11**.

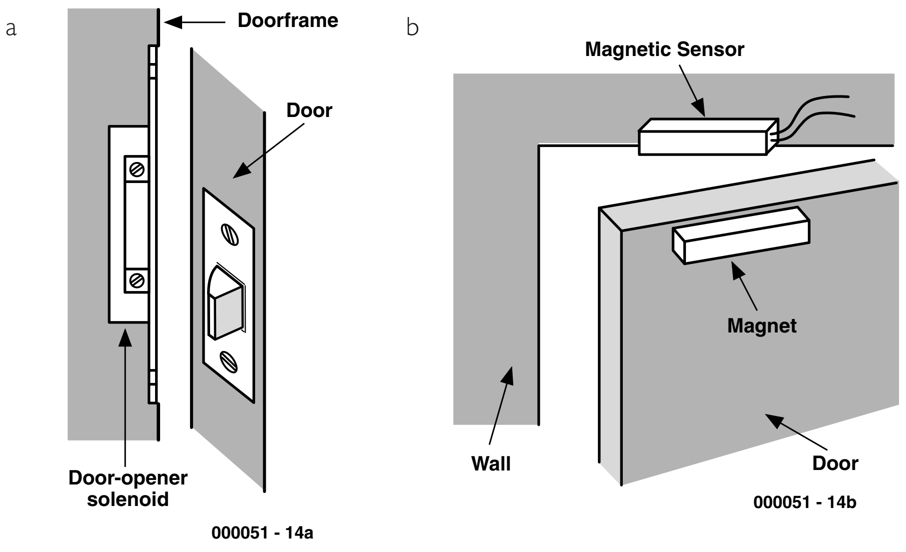


Figure 4. Suggested method of mounting the door opener solenoid and the magnet/reed switch combination.

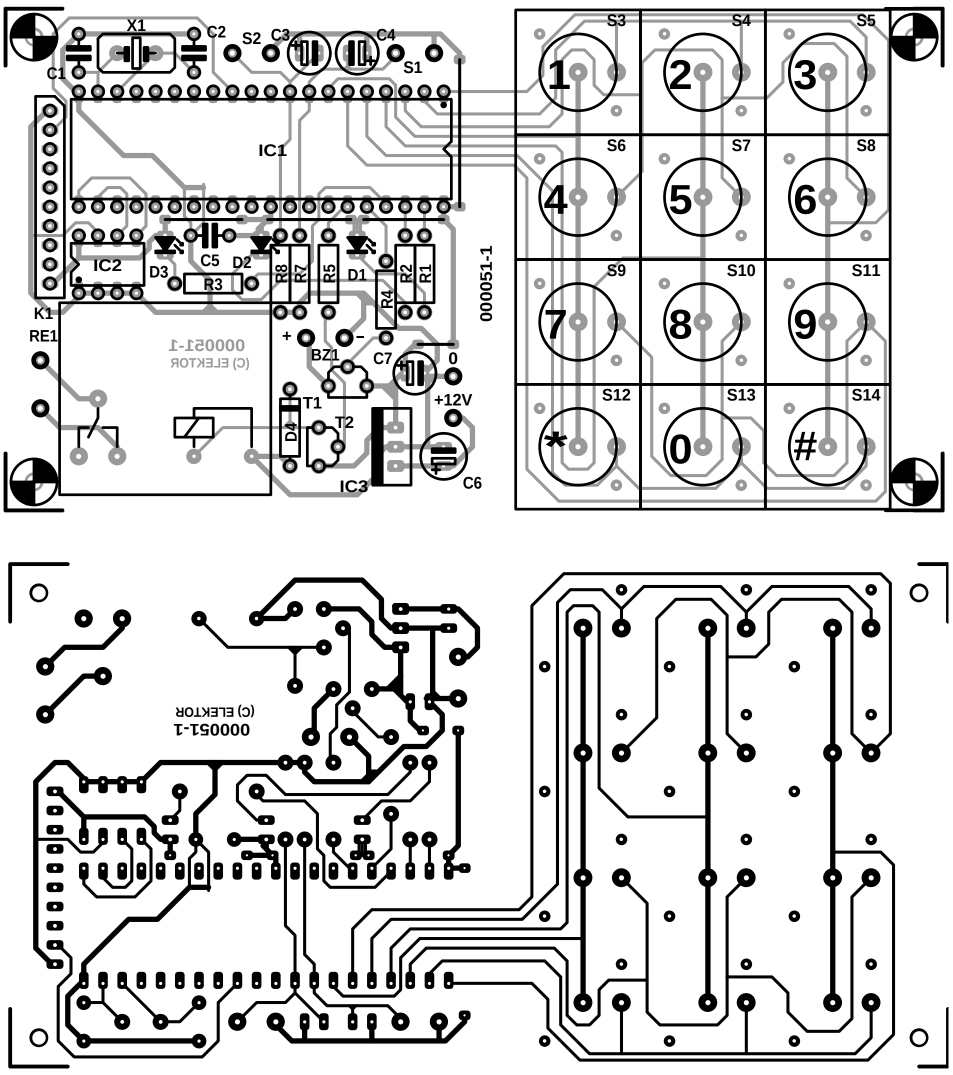


Figure 5. Printed circuit board layout. Board available ready-made through our Readers Services.

is unlocked and the latch is on there will be a brief time interval to allow the authorized user to enter.

DIY chip programming

You are lucky if you have a programmer for the AT89C52 micro, plus a suitable assembler. The assembly-code file (microcontroller program code) for the AT89 microcontroller used in this project is available either on disk (order code **000051-11**) or as a **free download** from the Subscribers Only section of our website http://www.elektor-electronics.co.uk.

Normal Operation Mode

This is the usual operation mode. The software flow diagram in **Figure 3** also shows the status of the various indicators and output devices when the microcontroller branches to certain routines depending on external conditions.

The main functions connected with the opening of the door are performed during normal-mode operation. The door-opener solenoid will be switched on only if the access code is correctly typed on the keypad. The 'door-open' alarm will also be activated when necessary.

The system will always operate in this mode unless it is rebooted to enter the supervisor mode.

The system will continue to be clocked while the door is open and a green LED indicator will show this state while the 'open-door' alarm is still on but the built-in delay is over.

Supervisor Mode

Only this mode allows the system codes to be changed. The only way to enter this mode is to hold the *, 5 and # keys down while starting-up the system. The system will confirm the positive result of the supervisor test by a beep from the buzzer and by flashing the three light indicators in the same rhythm. In addition, a time interval is created for the release of the *, 5 and # keys without considering any one of these as the first digit of a possible entrance code. It is also important to make sure that the door is closed before starting the password modification process. A period of 30 seconds is allowed to change the entrance code. If supervisor mode is successfully entered but no further action is detected, the system will enter normal mode after some time. This delay is reset after each entrance code modification.

The operations needed to activate or erase the different access entrance codes are the following. Firstly the supervisor mode has to be entered. Then, the password you want to add or erase is typed on the keypad. Finally you should push # to add the password, or * to delete it.

Reset

The system can be reset by pushing a key installed in the circuit. As a matter of course, this switch is not easily accessible. The password validated before the system is reset will be kept in memory.

Installation

A suggested installation of the door-opener solenoid and the magnetic sensor is shown in **Figure 4**. The ideal place to mount the magnetic (reed) contact is on the top part of the door frame (**Figure 4a**). The fixed magnet that belongs with the reed switch then becomes the 'mobile' part by attaching it to the top edge of the door as shown in **Figure 4b**. Door magnets and reed switches are usually sold by the pair.

Construction

The layout of the PCB designed for the project is shown in **Figure 5**. The sections reserved for the actual control circuit (left) and the keypad (right) are easily distinguished. Do not cut the board between these sections however, as you will break the seven interconnection tracks and no connector system has been provided as an alternative. As you can see from the introductory photograph, it is possible to use an existing keypad and wire it to the relevant solder pads on the keyboard section of the PCB.

There are no special precautions to observe when populating this board. Make sure, however, you know the required length of the LED terminals before soldering these devices in place. The aim is, of course, to ensure that the tops of three LEDs are level with the keytops so they can all protrude a bit from the top panel of the case. For the Atmel microcontroller we strongly suggest using an IC socket.

Provided the unstabilized input voltage from your mains adaptor does not exceed 12 V, voltage regulator IC3 should not require a heatsink.

Electrical installation

The connections to and from the main board have to be made such that the relevant wires can not be easily manipulated. Along the same train of thought, the controller/keyboard circuit has to be built in a tamper-proof enclosure. If you can not obtain the specified pushbuttons and caps, an alternative is to wire up an existing keypad and (if necessary) change the keycap symbols. We leave it to you to find ingenious ways of realizing these requirements! After all, you want to protect what's yours, don't you?

(000051-1)

Kinetic Auto Relay

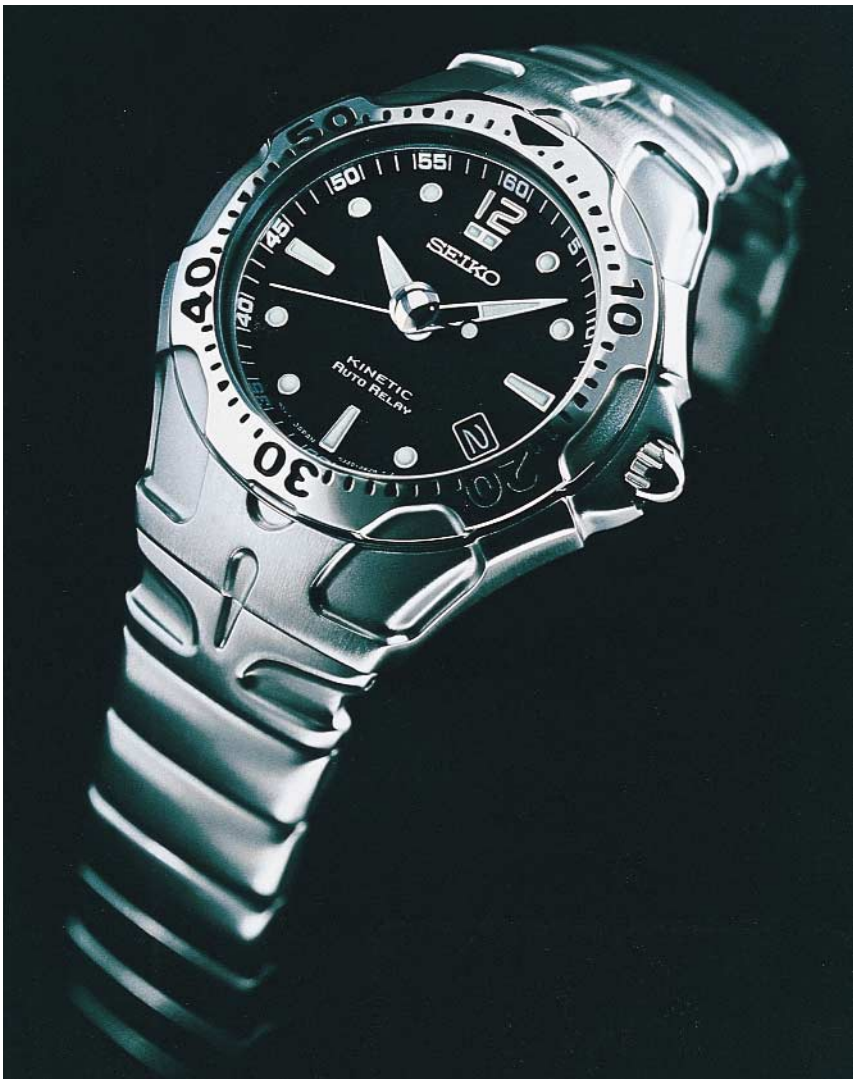
electromechanics by the square centimetre

By S. van Rooij

Seiko recently introduced a new line of wristwatches under the somewhat cryptic name of 'kinetic auto relay'. The internal construction of these watches will certainly interest electronics engineers. They are such magnificent examples of technical ingenuity that the fact that they also keep very exact time will probably be of secondary importance to the true enthusiast — even if it is naturally a nice side effect.

What do wristwatches have to do with electronics? Previously, the answer was 'nothing at all', but nowadays the two have steadily more in common. In any case, we can say that the boundary between the two disciplines is becoming increasingly vague, as a consequence of ongoing digitalisation and integration. A digital camera, for example, is a long way removed from its original, purely optomechanical roots. It is not at all a mechanical device any more, and it could just as well be described as a 'mini-computer with an optical interface'.

For some time now, a similar situation has prevailed with modern watches. Of course, this is obviously true of digital watches with liquid crystal displays, but even in the case of an analogue quartz watch, a traditional watchmaker would look in vain for his familiar gearwheels. A battery, an oscillator and a stepper motor are about all that you will find inside a modern quartz watch. This is all you need to make a watch that will keep accurate time, but it's not really an interesting piece of technology — certainly not for a watchmaker, but also not for an electronic engineer.



Automatic

The watch industry has not turned completely to quartz, in spite of appearances. One type of watch that has managed to survive very well, and that still has not yielded to the flood of quartz mechanisms, is the automatic watch. Such a watch can be recognised by the label 'automatic' on its dial.

An automatic watch is an old-fashioned type of watch, full of genuine gears, pawls and springs. It is called 'automatic' because it continuously winds itself from the movement of the wearer's wrist. This necessary energy comes from an eccentrically mounted weight that is coupled to the mainspring of the clockwork. The highly coveted Rolex still works on this principle, as do several other exclusive Swiss models.

You could of course decide to buy such an automatic watch for purely nostalgic reasons, since it is a nicely old-fashioned piece of technology. However, an automatic watch also has a true advantage with respect to a modern quartz watch, in that you will never be stuck with an empty battery at an inopportune moment. Against this stands the disadvantage that a mechanical watch can never be as accurate as a quartz watch.

Figure 1. In both 'automatic' and 'kinetic' watches, kinetic energy is converted into stored energy for operating the clockwork. No batteries needed!

Kinetic

A number of years ago, Seiko made a successful attempt to combine the advantages of automatic watches with those of quartz watches. They started with the same sort of eccentrically mounted moving weight that is the hallmark of all automatic watches. However, instead of using this to wind a spring, they used it to drive a miniature generator. This was very clever on the part of the developers, since the generator was used to charge a battery or a capacitor, which in turn supplied power to — you guessed it — a quartz movement! In this way, they created an automatic watch that can offer the high accuracy of a quartz watch. In addition, the combination of a generator and a capacitor turns out to yield a much greater reserve capacity than the spring of a classic automatic watch. Normally, a spring cannot drive a watch for more than one or two days if the watch remains stationary, but the generator can build up a reserve that can power the watch for several weeks. Seiko gave the new drive technology the name 'kinetic'. Over the last few years, watches based on this principle have developed into a reasonably successful alternative to traditional automatic watches.

Auto-relay

With the recent development of the 'kinetic auto-relay' watches, Seiko has once again shown that they are a trend-setter in modern watch technology. These watches contain an additional, intelligent control chip, which allows the energy reserve time to be extended to at least four years. If the watch remains stationary for three days in a row, this chip causes the hands of the watch to be stopped. A soon as the watch is picked up again after such a dormant period, its hands are automatically set to the correct time.

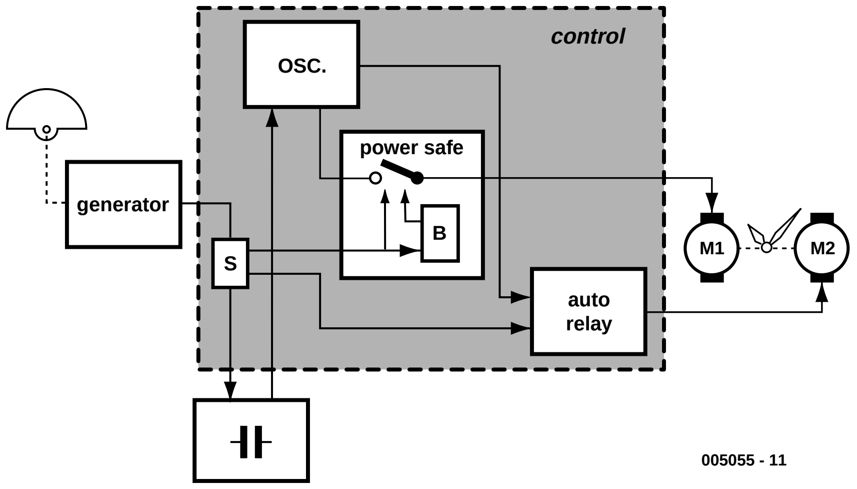


Figure 2. Simplified block diagram of the 'kinetic auto-relay' system.

The idea behind this new development is actually quite simple. At some point, Seiko realised that the actual clockwork (the oscillator) consumes only 15 percent of the energy needed to run the watch, while the remaining 85 percent is used to drive the hands. This means that if the reserve time is to be increased, the means to do so must be sought in the drive mechanism for the hands.

This is however easier said than done. Stopping the hands is naturally not all that difficult, but setting them to the correct positions after a dormant interval is another story. Still, this problem has been solved with the help of a clever controller and an extra stepper motor.

Figure 2 shows a simplified block diagram of the auto-relay system. The generator (plus the rotating weight), the quartz crystal oscillator, the capacitor or battery used to store the energy and the stepper motor (M1) are all elements that are used in the 'standard' kinetic system. The new elements are the sensor (S), the buffer (B), a second stepper motor (M2) and the 'power safe' and 'auto relay' blocks.

In general, the system works as follows. When the watch is being worn, so that the rotating weight occasionally moves, the generator delivers a charging current to the capacitor more or less regularly. The capacitor provides the oscillator with its operating voltage, so that the oscillator can drive stepper motor M1, which in turn drives the hands.

If the watch does not move, no charging current is generated. Sensor S (which in principle is just a shunt resistor) detects this fact and generates a command. This causes the 'power safe' block to de-assert the enable signal for the switch that connects the oscillator to motor M1. However, this does not have an immediate effect, since the buffer B holds the switch enabled for 72 hours. At the end of this time, the control signal for M1 is interrupted and the hands stop moving.

When the watch is moved again after some time, the generator will again produce a charging current. The sensor detects this and passes this information on to the 'auto relay' block. This block compares the position of the hands to the current time, as provided by the oscillator. This block then positions the hands to the correct time, moving the hour and minute hands first and then the seconds hand. The only thing that has to be set manually is the date display (if one is present), since automatically setting this would consume too much energy.

The normal hand drive mechanism (with motor M1) is also re-enabled via the 'power save' block, so that the watch again runs normally.

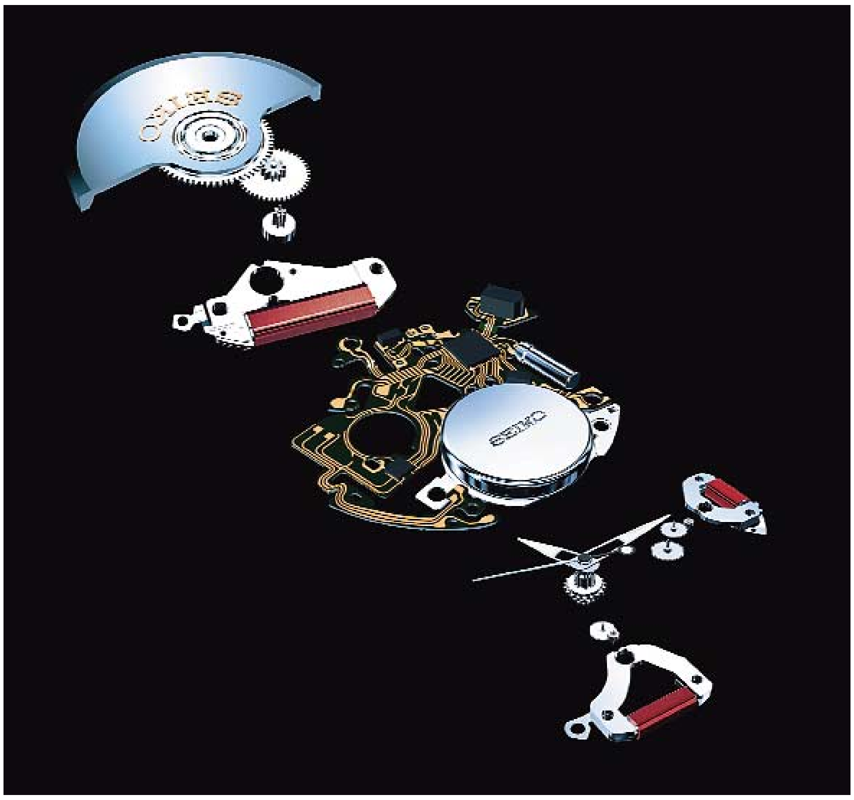


Figure 3. Exploded view of the interior mechanism of the watch. The chip located on the circuit board controls all functions of the watch.

Internal details

A block diagram of how the watch works is naturally nice, but a look inside the actual mechanism is much more informative. The internal elements of the new 'auto-relay' mechanism are shown in some detail in **Figure 3**. For clarity, the various elements have been spread out in this illustration, since it's difficult to see how they are built when everything is assembled in its working configuration.

At the top, you can see the rotating weight, which weighs around five grams. It is mounted in low-friction bearings, so that it changes position every time the wearer's wrist moves. The rotating weight drives a minuscule rotor via a two-stage gear train. The motion of the rotor induces a current in the stator coil, which is shown below the rotor. The rotor is not much larger than the head of a pin, and it can spin at up to 100,000 rpm.

Below this, you can see the printed circuit board, which has a cutout in the middle for the controller chip. The quartz crystal for the oscillator is located on the right hand edge of the circuit board, and the 'energy storage unit' is clearly visible below the circuit board. Previously, a capacitor was use for energy storage, but the specifications presently provided by the manufacturer suggest something more like a rechargeable battery.

Finally, at the very bottom, you can see the hands and the two drive motors.

Working reserve

The energy storage unit (the battery) is delivered fully charged from the factory. If the power reserve somehow becomes used up, due the watch not being used for a long time, the watch will naturally have to be worn for a while to build up a sufficient power reserve. A reserve that will last for a few days or a week can be built up very quickly, but it takes a certain amount of time to accumulate the maximum reserve capacity (four years). If the watch is worn every day for 12 hours a day, the maximum working reserve will have been built up after 168 days.

(005055-1)

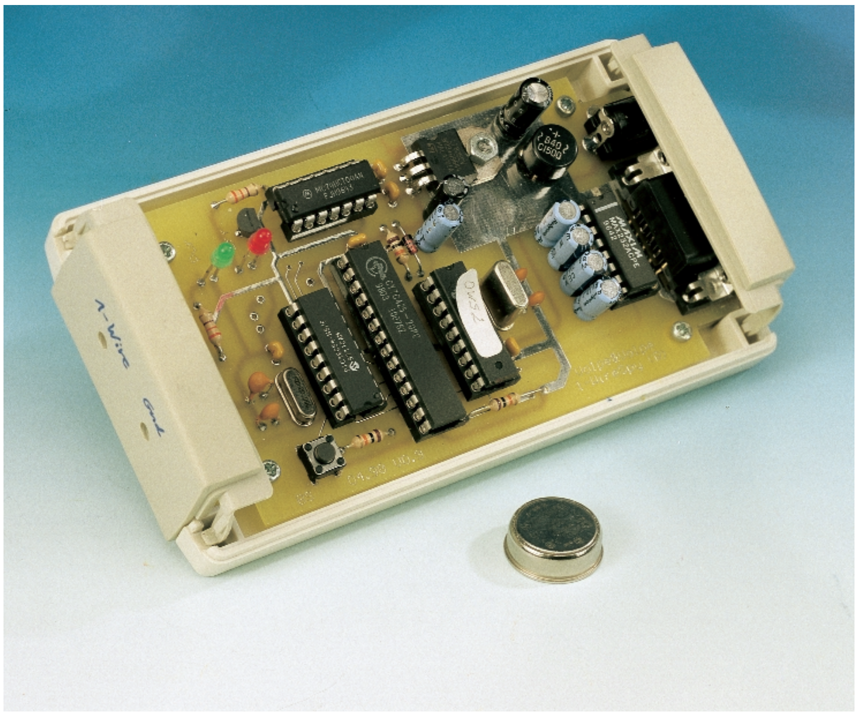
One-Wire Spy

monitor for 1-wire ICs

Design by Prof. Francesco P. Volpe – Email volpe@fh-aschaffenburg.de

One-Wire Spy is a useful aid that enables a PC to monitor the communications between a microcontroller and a 1-wire IC from Dallas Semiconductor.

The semiconductor manufacturer Dallas Semiconductor offers a series of components that can be controlled, read and programmed by a master device (such as a microcontroller) via a single-wire bus. These '1-wire' devices are mainly available as temperature sensors, battery-management ICs and memory ICs. There is also an identity IC, which contains only an identification serial number. **Table 1** summarises the available devices. With a 1-wire bus, it is possible to use not only normal IC packages, but also TO-92, PR-35 and flip-chip packages. Dallas Semiconductor even offers ICs in button-cell format, under the name 'iButton'.



Data transfer

Data are transferred to and from a 1-wire device using the 1-wire protocol. An integrated multiplexer decodes the incoming signals and ensures that the chip receives the correct signals.

Each 1-wire device has an open-drain output driver, which allows several 1-wire devices to be connected to a single bus in a wired-AND circuit with a shared resistor. It also allows all sorts of normal hardware to be connected to the bus. For this reason, Dallas Semiconductor calls this bus the 'microLAN'.

Dallas offers 1-wire devices with SRAM, EPROM, and EEPROM memory, addressable switches and complete battery management ICs. All types have 8 bytes of ROM, in which a family code, a six-byte serial number and a CRC-8 byte (cyclic redundancy check byte) are stored. This is not a mask-programmed ROM. Instead, the eight bytes of information (including the serial number) are written to each chip at the end of the fabrication process by using a laser beam to selectively cut polysilicon tracks. This means that all chips of a given type can be produced using the same, very expensive mask set.

Commands and data are transferred to and received from a 1-wire device one bit at a time until the command is fully completed, with the lsb first. The microcontroller is the master in the data transfer process, and the 1-wire device is the slave. The master establishes the bit synchronisation by placing a high-to-low transition on the data line. After this, either the master or slave samples the data line, according to the command and the transfer direction. Each bit is individually synchronised during the data transfer, which means that irregular data transfers with pauses are certainly allowed.

Reading and writing takes place in 'time slots'. Normally, the master sends a Reset (t_{RSTL}, low for at least 480 μs) to create a defined initial condition. This is followed by an equally long interval with the data line held high. During this interval (t_{RSTH}), the 1-wire device generates a Presence pulse, which allows the master to determine whether a 1-wire device is connected to the bus (see **Figure 1**).

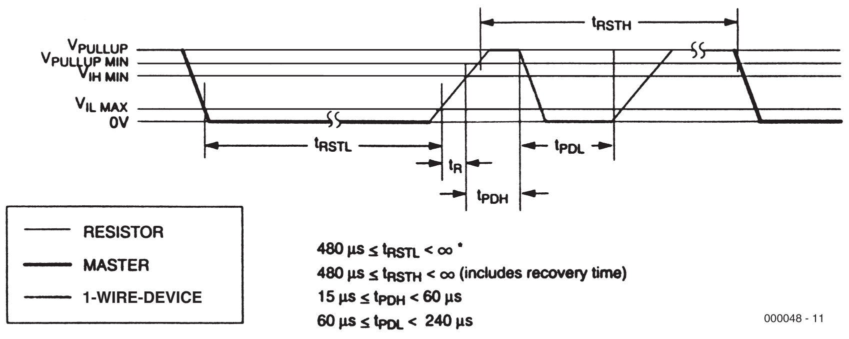
After the master has initiated the transfer with a High-to-Low transition, data bits can be transferred to the 1-wire device in write time slots that are 60 to 120 μs long. The 1-wire device samples the data line 15 to 60 μs after the falling edge of the synchronisation signal. If it finds a High level, it interprets this as a 1 (see **Figure 2**), while a Low level will be interpreted as a 0 (see **Figure 3**). Due to the timing tolerance of 15 to 60 μs, the data line must be held stable during this interval. Following this, the 1-wire device needs a recovery time of at least 1 μs before the next bit can be sent.

The master reads data in a similar manner (see Figure 4). Once again, the master starts with a synchronisation signal (falling edge), following which the 1-wire device sends one bit from the addressed memory location. If the bit is a 1, the 1-wire device does not have to do anything, since the pull-up resistor will hold the data line at a High level. With a 0, by contrast, the device pulls the data line Low for 15 μs. The master can sample the line during this interval. The master must hold the line Low for at least 1 μs after the synchronisation falling edge. However, it should not hold the line Low any longer than this, in order to keep the sampling window as large as possible. After the bit has been transferred, the 1-wire device needs between 0 and 45 μs to release the line.

Commands and data are transferred to the 1-wire device by a series of write-zero and write-one time slots (see **Figure 5**). To read data and receive return messages, the master must generate an appropriate number of read-data time slots.

Wiretapping

The overall arrangement for reading and writing data with 1-wire devices is very intricate. This means that it is probably 'normal' that newly generated software will have errors, which could be very difficult to correct without a record of the data stream. The One-Wire Spy generates a log of the data transferred over the 1-wire bus. It is simply connected



t_{RSTH}
V_{PULLUP}
$V_{PULLUP\ MIN}$
$V_{IH\ MIN}$
$V_{IL\ MAX}$
0V
t_{RSTL}
t_R
t_{PDL}
t_{PDH}
RESISTOR
MASTER
1-WIRE-DEVICE
$480\ \mu s \le t_{RSTL} < \infty$ *
$480\ \mu s \le t_{RSTH} < \infty$ (includes recovery time)
$15\ \mu s \le t_{PDH} < 60\ \mu s$
$60\ \mu s \le t_{PDL} < 240\ \mu s$
000048 - 11

Figure 1. The master must hold the data line Low for at least 480 μs to generate a Reset pulse. The 1-wire device responds with a Presence pulse.

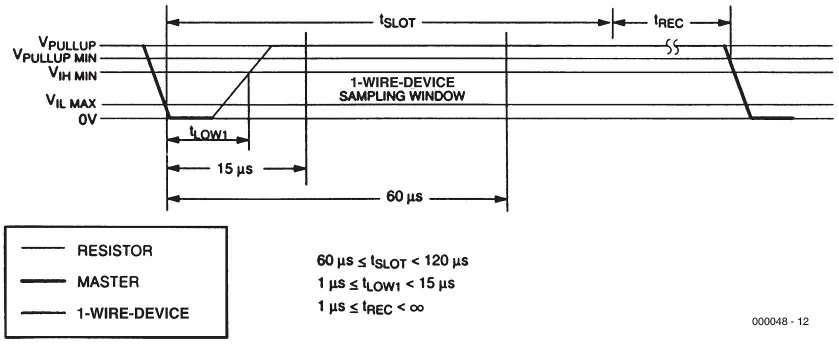


Figure 2. The master writes a 1 by releasing the data line after the negative edge, with a maximum delay of 15 μs.

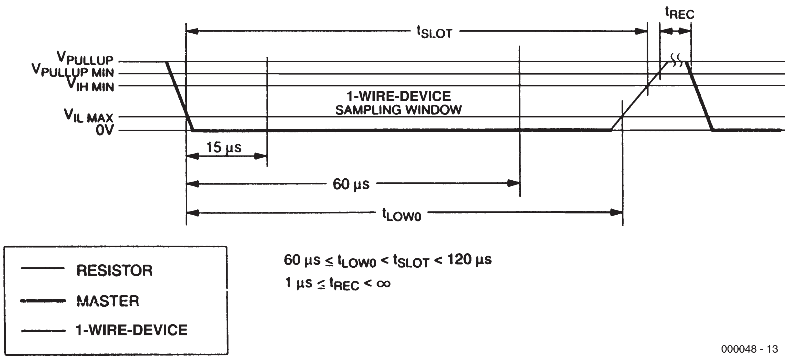


Figure 3. The master must hold the data line Low for at least 15 μs to write a 0.

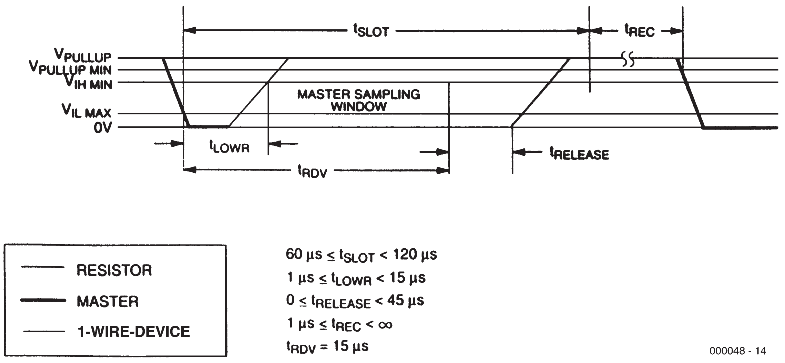


Figure 4. The master samples the data line within 15 μs following the negative edge to read data from the 1-wire device.

between the master and the 1-wire device, as shown in **Figure 6**, and then connected to a PC via an RS232 interface. A valid Reset pulse is necessary to trigger the One-Wire Spy. After this, it is synchronised to the following data stream. Depending on whether a 1-wire device is connected to the bus, the One-Wire Spy sends either the ASCII string 'RS' (no device) or the string 'RP' (1-wire device properly connected to the bus). It also sends all received bytes as ASCII strings. The data transfer between the One-Wire Spy and the PC follows the serial RS232 standard at 9600 baud, with 8 data bits and 1 stop bit.

The data rate on the 1-wire bus can be up to 16.3 kbit/s, but the One-Wire Spy can transfer data to the PC at only 9.6 kbit/s. This means that the One-Wire Spy must buffer the data. A fast FIFO (first in, first out) memory can be used for this purpose. For normal applications, a memory size of 1 kB is adequate, but sizes of up to 4 kB can readily be used.

Based on the above description, we can assign tasks to the two microcontrollers shown in the circuit diagram (**Figure 7**). The bus is connected to one port of microcontroller IC3 (PIC16C54). This IC converts eight serial bits into a parallel byte, and passes it to the FIFO memory via a write instruction (WRFIFO).

The second microcontroller is a PIC16F84 that converts the parallel bytes back to a serial data stream, but this time according to the RS232 standard. The TTL-level outputs of this IC are converted to symmetric ±12 V levels by the following MAX232 (IC2). The microcontroller can detect an overflow of the FIFO, and if this occurs it blocks the input microcontroller via the RPIC line. LED D2 is illuminated at the same time, but it remains illuminated only until the FIFO is once again ready to accept data. Any data that are transferred over the bus during this interval will not be logged.

Construction and installation

The One-Wire Spy can be built on the double-sided printed circuit board shown in **Figure 8**. This is unfortunately not available from Readers Services. However, the circuit board, as well as construction kits and fully assembled boards, may be obtained from Elektronikladen (*www.elektronikladen.de*). Note that this could not be ascertained at the time of writing this article.

The power supply is included on the circuit board. It consists of a bridge rectifier, a 5-V voltage regulator and a few capacitors. The green LED acts as a power-on indicator. Power is provided by a mains adapter that can deliver a DC voltage between 9 and 15 or an AC voltage between 7 and 12 V. This is connected to the 2.1-mm low-voltage jack BU2.

The construction of the circuit is not difficult, especially since all ICs may be fitted in sockets. The installation of the One-Wire Spy is equally simple. Start a terminal emulator program (such as HyperTerminal) on the PC, set the serial port parameters to 9600 baud, 8 data bits, no parity and one stop bit, and connect the One-Wire Spy to the serial interface connector of the PC via an RS232 cable. Now connect the mains adapter, which should cause the green LED to light up. The terminal

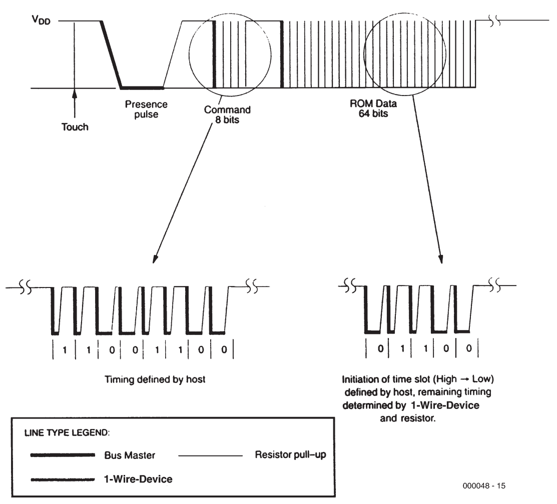


Figure 5. Following the Presence pulse from the 1-wire device, the master transfers a command to read the ROM. The 1-wire device responds with the content of its ROM.

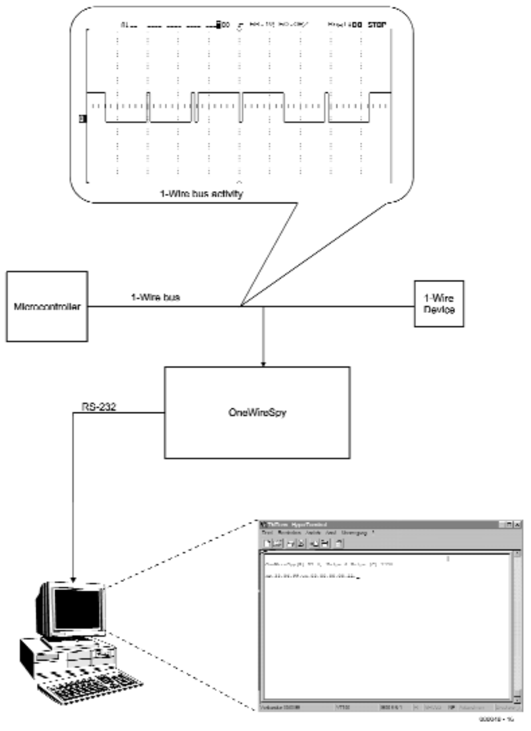


Figure 6. Coupling a 1-wire device to the 1-wire bus.

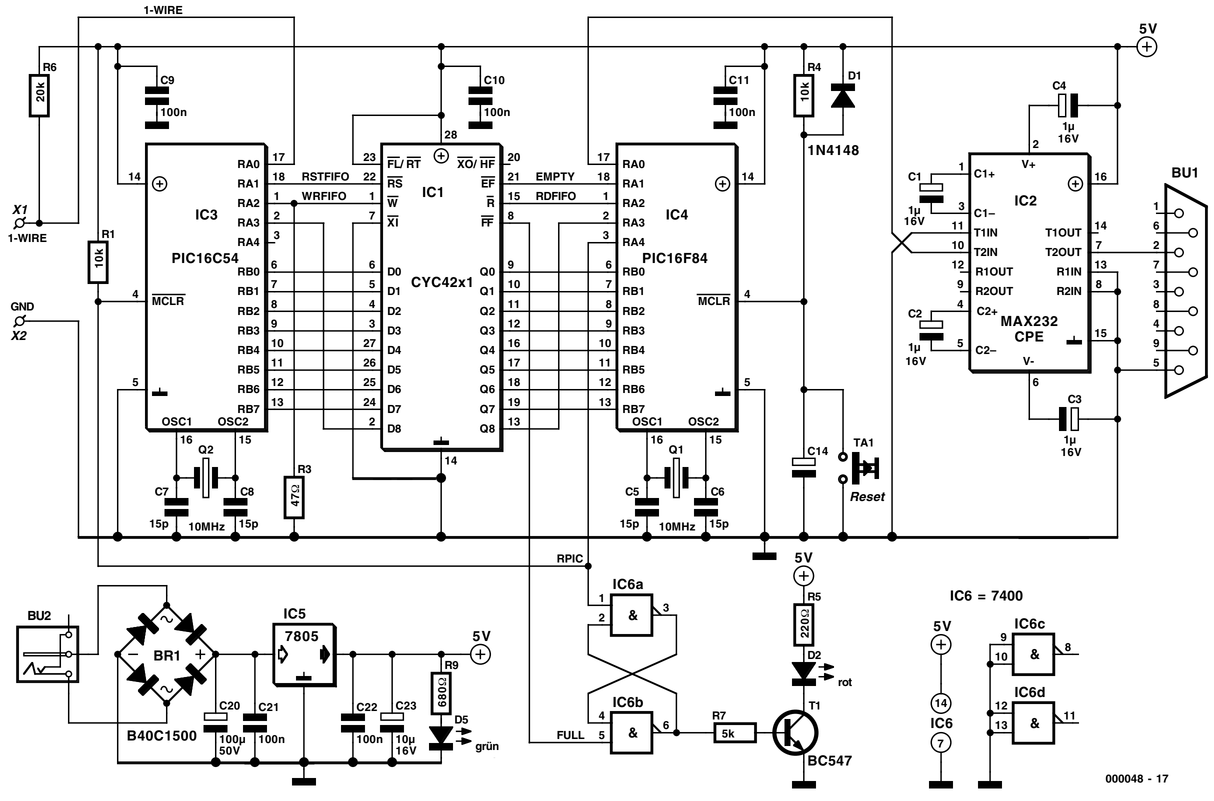


Figure 7. The hardware of the One-Wire Spy consists of a FIFO memory and two microcontrollers.

Table 1. Summary of 1-wire devices available from Dallas Semiconductor.

1-Wire-Device	Function	Memory
DS1820	Digital Thermometer	16 Bits EEPROM
DS18B20	Programmable Resolution Digital Thermometer	16 Bits EEPROM
DS18S20	High-Precision Digital Thermometer	16 Bits EEPROM
DS1821	Stand-alone Thermostat	2 bytes NV
DS1822	Programmable Resolution Digital Thermometer	No NV
DS2401	Silicon Serial Number	No additional memory
DS2404	EconoRAM Time Chip	4096 bits RAM
DS2404S-C01	Dual-Port Memory Plus Time	4096 bits RAM
DS2405	Addressable Switch	No additional memory
DS2406	Dual Addressable Switch	1024 Bits EPROM
DS2409	MicroLAN Coupler	No additional memory
DS2417	Time Chip with Interrupt	32-bit Real Time Clock Counter
DS2423	1-Wire RAM with Counters	4096 bits RAM
DS2430A	1-Wire EEPROM	256+64 bits EEPROM
DS2433	1-Wire EEPROM	4096 bits EEPROM
DS2434	Thermometer	32 byte EEPROM, 32 bytes SRAM
DS2435	Thermometer/Time-Temperature Histogram	32 bytes EEPROM, 32 bytes SRAM
DS2436	Thermometer, Voltage A/D	32 bytes EEPROM, 8 bytes SRAM
DS2437	Fuel Gauge, Voltage A/D, RealTime Clock, Temperature	40 bytes EEPROM
DS2438	Fuel Gauge, Voltage A/D, Elapsed Time, Temperature	40 bytes EEPROM
DS2450	1-Wire Quad A/D Converter	Status Control Memory Only
DS2480B	1-Wire Line Driver	Status Control Memory Only
DS2490	USB to 1-Wire Bridge	Chip Mode Control and I/O FIFOs
DS2502	Add-only Memory	1024 Bits EPROM
DS2502-UNW	UniqueWare	1024 Bits EPROM
DS2502-E64	IEEE EUI-64 Node Address Chip	256 bits pre-programmed, 768 bits user-programmable
DS2505	Add-only Memory	16,384 Bits EPROM
DS2505-UNW	UniqueWare	16,384 Bits EPROM
DS2506	Add-only Memory	65,536 Bits EPROM
DS2506-UNW	UniqueWare	65,536 Bits EPROM
DS2890	1-Wire Digital Potentiometer	Feature and Control Memory
DS9502	ESD Protection Diode	–
DS9503	ESD Protection Diode with Resistors	–

emulator will report the startup message 'OneWireSpy' (possibly after you press the Reset button), and after this the One-Wire Spy waits for data on the bus.

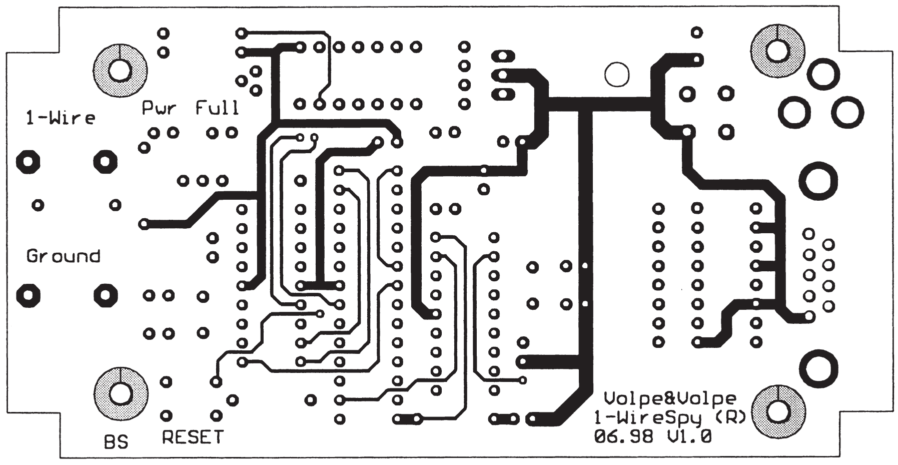
(000048-1)

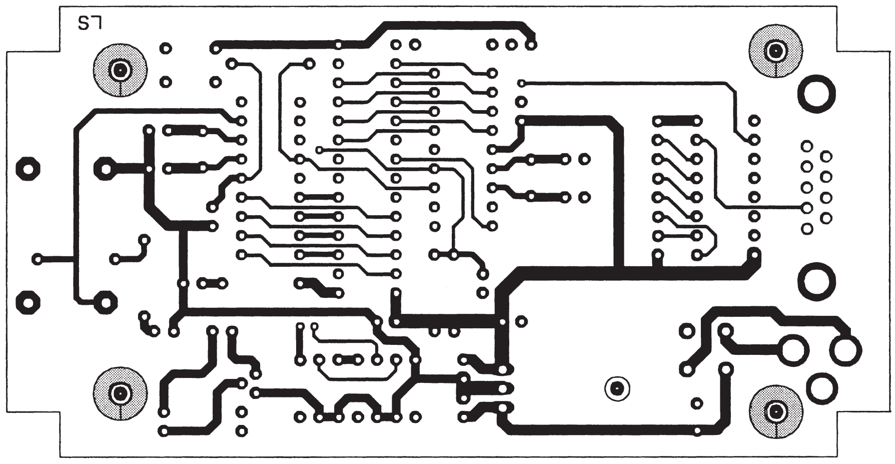
Further information on iButton chips from Dallas Semiconductor:
www.dalsemi.com
www.ibutton.com

On Project Disk # 000048-II

Hex16-Listings of both PIC programs
PCB layout, Eagle file
PCB layout, TIF file

Internet:
Project software # 000048-11 may also be downloaded from our website at http://www.elektor-electronics.co.uk (subscribers only).





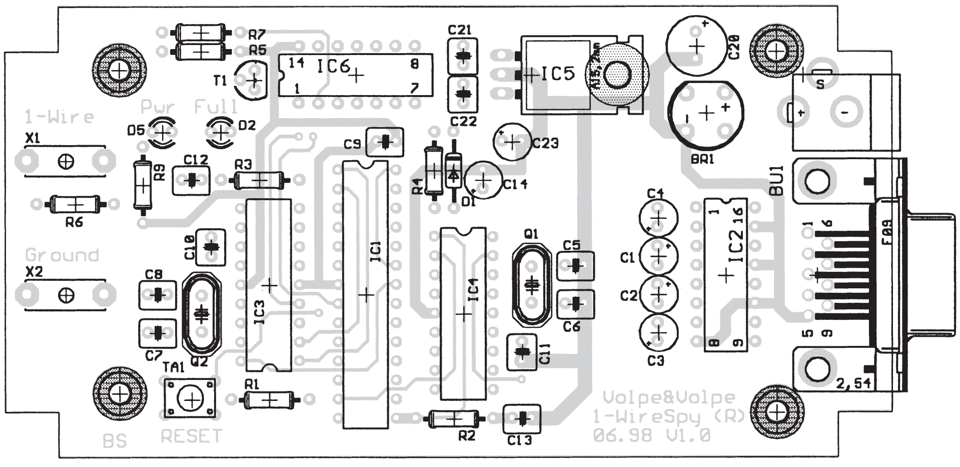


Figure 8. The double-sided printed circuit board for the One-Wire Spy (not available from Readers Services).

COMPONENTS LIST

Resistors:
R1,R4 = 10kΩ
R3 = 47Ω
R5 = 220Ω
R6 = 20kΩ
R7 = 5kΩ
R9 = 680Ω

Capacitors:
C1-C4 = 1μF 16V radial
C5-C8 = 15pF
C9,C10,C11,C21,C22 = 100nF
C14 =
C20 = 100μF 50V
C23 = 10μF 16V

Semiconductors:
D1 = 1N4148
D2 = LED, 3 mm, red
D5 = LED, 3 mm, green
Br1 = B40C1500 (40V piv, 1.5A peak)
T1 = BC547
IC1 = CY7C42x1 (Cypress Semiconductor)
IC2 = MAX232CPE (Maxim)
IC3 = PIC16C54 (programmed, order code **000048-42**)
IC4 = PIC16F84 (programmed, order code **000048-41**)
IC5 = 7805
IC6 = 74HCT00

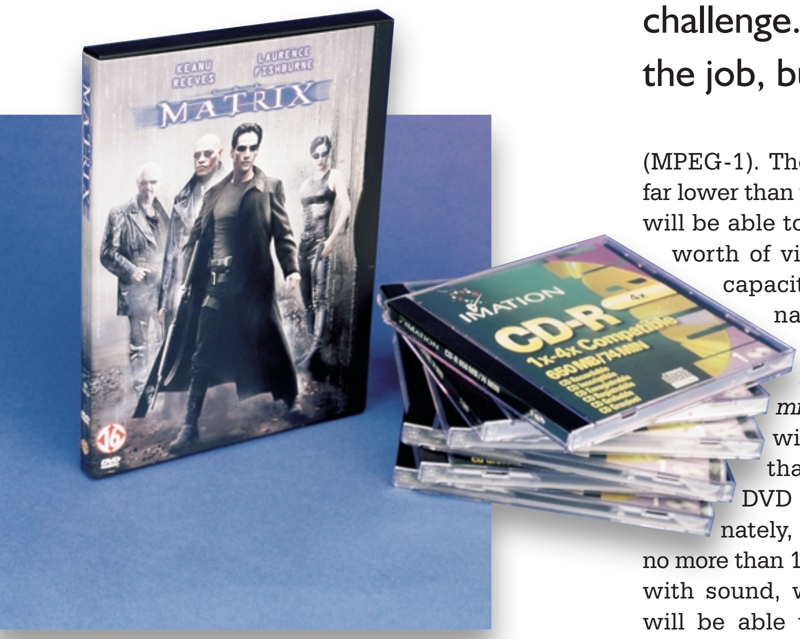
Miscellaneous:
X1,X2 = 10 MHz quartz crystal
S1 = miniature pushbutton, 1·make contact
Bu1 = 9-way sub-D socket (female), PCB mount
Bu2 = Mains adaptor socket, PCB mount, 2.1 mm
Sockets for all ICs (except IC5)
Disk, order code **000048-11**

Make Your Own Mini DVDs

15 minutes worth of film on a CD-R

By Harry Baggen

The fact that DVDs are not easily duplicated makes just about any attempt at copying these discs a real challenge. A number of programs are available to do the job, but as you will see it is far from easy.



The tremendous amount of data coupled with the complex way image and sound are interwoven, and not forgetting the built in copy protection system make it pretty difficult to copy programme material stored on DVD. As we write this, there are no affordable DVD recorders and blank media. None the less, many DVD users of the enthusiastic type have been able to copy favourite scenes from DVD movies to hard disk or CD-ROM.

In principle, there is no objection against copying film fragments for your own personal use, for example, to compile your own collection of film or video clips. By means of a variety of protective measures including the region code and CSS, it is not possible with most DVDs to copy a VOB file, that is, the file containing the picture and sound information) to hard disk. And even if you manage to obtain a usable VOB file, there are more hurdles, because a fair number of operations are required to arrive at a file containing the desired video and sound information. We went through the trouble of reading some of the relevant descriptions that circulate on the Internet. There exist two popular variants. One s to convert DVD information in such a way that a video CD can be produced (MPEG-1). The resultant quality is far lower than that of a DVD, but you will be able to squeeze 60 minutes worth of video into the storage capacity offered by an ordinary CD-R. The other method boils down to making a so-called *mini-DVD*. This is a CD-R with the file structure that complies with the DVD standard. Unfortunately, such a CD-R will hold no more than 10 to 15 minutes of film with sound, while not every DVD will be able to reproduce such a mini-DVD correctly.

In both cases you start with copying the VOB file to hard disk. This is usually done with 'crack' utilities like DeCSS or VOB Ripper. By killing the copy protection code, these programs ensure that the VOB file can actually be used. Although the film industry will do just about anything to get these programs off the face of the earth, that is nearly impossible to achieve as long as they are used for private purposes.

If you want to make a video CD, the next step is to convert the picture format from NTSC to PAL (if necessary) using a utility like NTSC2FILM. Next, the result may be converted into video-CD format using an MPEG-1 encoder such as the Panasonic encoder with associated VOB filters. Most encoders are unable to tackle VOB files directly. Worse, a complex intermediate step is required to parse the VOB files into its sound and picture components, which subsequently have to be combined again into another format like AVI.

Making a mini-DVD is even more elaborate because you also have to create your own menu structure.

The steps to follow are basically the same as described above. In this case, however you have to employ an MPEG-2 encoder to create a new MPEG2 file. Next, the new M2V and AC3 sound file have to be multiplexed (or merged) into one file. This is usually done with programs like Streamweaver. The required menu structure for the mini-DVD is usually produced with a like CDMotion, which allows your own opening menu to be designed with start buttons for the various scenes, etc. The end result will be a directory called Video_ts containing all files you will need to make a DVD. Using a program like Nero, the whole caboodle may then be burned onto a CD-R. If everything went well, you then have a mini-DVD (well, actually a CD-R with 15 minutes worth of image and sound). In practice, we found that it took about 10 hours of computer time to copy a film of about 1.5 hours on a number of CD-R discs.

That, admittedly, is a lot of trouble to go through for a film you can buy for about £15. Very few people will be prepared to go to such lengths to copy DVD material as described here. So, the film industry has little to worry about in the absence of affordable DVD recorders, very fast computers and, of course, blank disks. In essence, the methods and programs mentioned here are for DVD freaks only and possibly for criminals involved in large-volume production of illegal DVD copies.

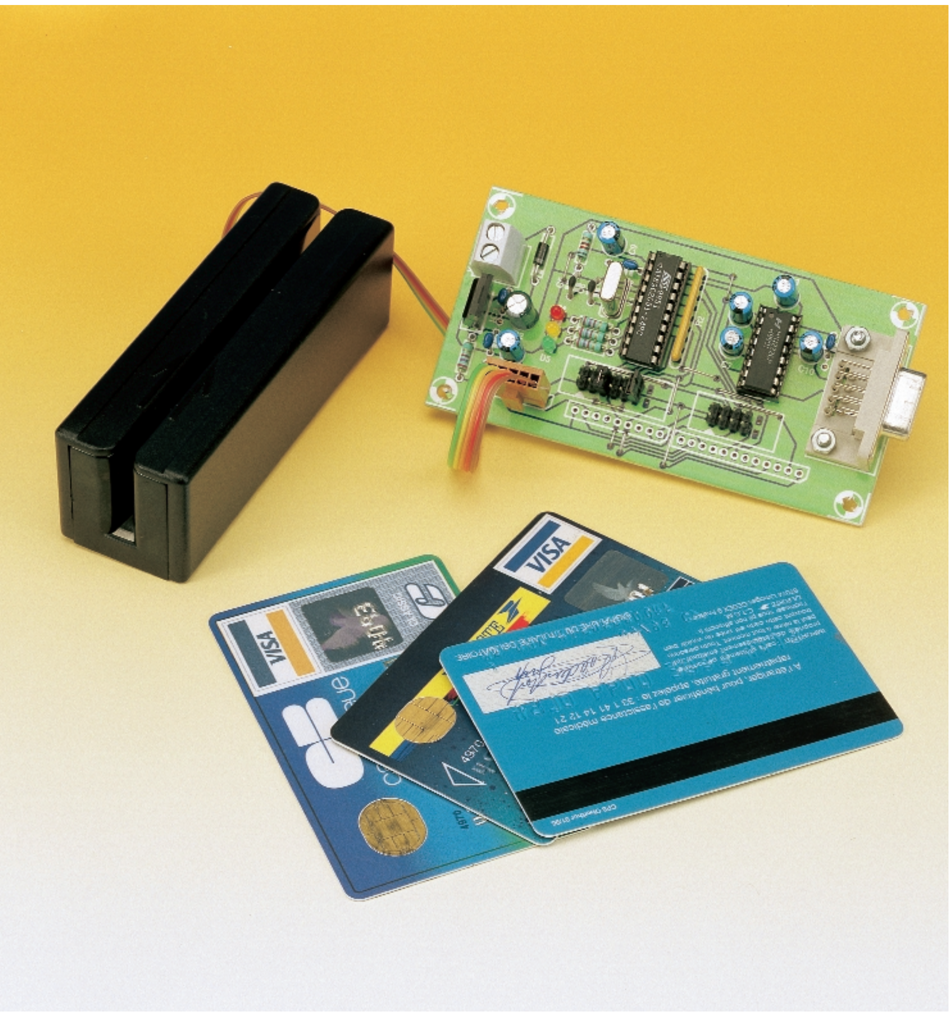
(000056-1)

Magnetic Card Reader

tracking down the ISO tracks

By Karo Bauer

Credit cards, Eurocheque cards, other types of bank cards and car park tickets: even if many of these cards contain chips, they all have magnetic strips. This article describes how you can read and decode the contents of these magnetic strips.



Our project consists of three parts: a magnetic-strip ('swipe') card reader for 'playing back' the data tracks, an interface that connects the card reader to the serial port of a PC, and finally, Windows software for decoding, displaying and managing the data read from the cards.

Swipe reader

A standard commercial 'swipe' (pull-through) reader is used to read the cards. With this device, the card is manually pulled through a slot and thus across a read head. In principle, this works exactly the same way as every magnetic tape (cassette) data recorder, except that here the 'tape' is only a few centimetres long and is glued to the card. In fact, if you want to make your own card, you can cut a suitable length of magnetic tape from a cassette (use a VHS cassette for the proper width) and glue it to a plastic or cardboard card. However, in this case you will also have to put together your own magnetic card writer, or find one in an electronics surplus shop. Such devices are normally only available at 'professional' prices (in the four figures), while simple swipe readers can be found starting at around 20 pounds.

Even the simplest swipe reader contains a small circuit board with the playback electronics (amplifier and pulse shaper), and it provides a fully processed TTL signal at its

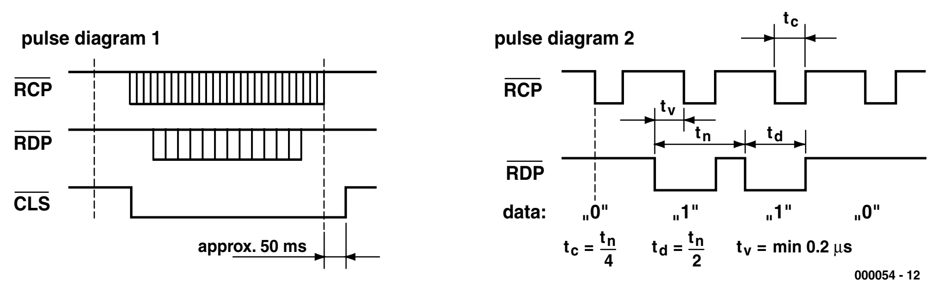


Figure 1. The output of the reader module delivers three signals for each track that is read. These are shown in two waveform diagrams.

ISO standard magnetic strip card coding

The bits are recorded on the magnetic tracks using the Aiken biphase method. The write head is driven by a pulse signal. The direction of the current through the head coil changes each time the polarity of the signal changes (at each transition). This causes the direction of magnetisation on the tape to also change. Each bit is written in a 'bit cell', which is a segment of the tape in which a logical '0' is represented by a single change in the magnetic polarity, while a logical '1' is represented by two changes in the magnetic polarity. This means that a single pulse is used to write a '0', while a pair of pulses is used to write a '1'. The pulses for a '1' are only half as long as the pulse for a '0'. The bit cells thus have the frequency of the '0' signal, and the '1' signal has exactly twice this frequency. This simple recording method has the advantage that a read clock signal can easily be generated from the read data signal, so that the bits can be decoded independent of the reading speed. A '1' always has twice the frequency of a '0', regardless of how fast the card passes by the read head.

The ANSI/ISO standard (ISO 3554) is predominantly used for recording and coding data on magnetic-strip cards. According to this standard, three tracks are recorded on the magnetic strip, each of which is 2.8 mm (0.11 inch) wide. If you hold the card so that the magnetic strip is horizontal and at the bottom, then track 3 is at the top of the strip, track 2 in the middle and track 1 at the bottom. If you look at the tracks, the data are read from left to right, just like the letters in a line of text. According to the ISO standard, each track has its own coding and intended use, as follows:

track 1:	210 bpi, 7-bit alphanumeric; 79 characters
track 2:	75 bpi, 5-bit BCD; 40 characters
track 3:	210 bpi, 5-bit BCD; 107 characters

This means that track 1 is the only track with text coding (such as the name of the cardholder). The coding of the data in track 1 uses the ANSI/ISO ALPHA data format, which is also called the ISO 7-bit format for short. For tracks 2 and 3, the ANSI/ISO BCD data format (ISO 5-bit format) is used for coding.

The ISO 7-bit format encompasses 64 different characters, using six bits per character. The least significant bit (lsb) comes first and is thus read first, while the sixth bit is the msb and the seventh bit is an odd parity bit. Of the 64 encoded characters, 43 are alphanumeric characters, 3 are boundary characters (start character, separator and end character) and 18 codes are used for control and special characters.

The ISO 5-bit format uses the first four bits to encode 16 different characters, while the fifth bit is again an odd parity bit. The first bit is the lsb, which is read first. Of the 16 encoded characters, 10 are numeric (the numerals), 3 are again boundary characters and the remaining 3 are control characters.

Recording data on a track of the magnetic strip always starts with a series of logical zeros ('0' bit cells), in order to synchronise the clock signal for the decoding. A start character marks the beginning of the actual data, so that the decoding circuit in the reader starts at the right place in counting off bits in groups of five or seven during the decoding. An end character stands at the end of the series of data bits in the track, and it is followed by an LRC character. 'LRC' stands for 'longitudinal redundancy check', which is a parity check bit for the sum of all the data bits of all preceding characters. This 'longitudinal parity' allows a double bit error in a character to be detected. This is not possible with the odd parity bits, since two bit errors within a single character balance each other out in the parity bit.

output. In fact, there are three signals for each track that is read, as illustrated in the two waveform diagrams shown in **Figure 1**. These are the clock signal, the data signal and a Ready signal. As you can see from the first diagram, the Ready signal (~~CLS~~) marks the start and end of the reading process when the card is pulled through the reader. The end point (when ~~CLS~~ returns to a High level) is delayed 50 ms from the last clock pulse. The second diagram shows the relationship between the clock signal (~~RCP~~) and the data signal (~~RDP~~). The value of the data signal (1 or 0) is sampled on the negative edge of the clock signal. The data signal is output in inverted form, so a Low level at the sampling time means a '1', while a High level means a '0'. The 'g' that marks the sampling point in the diagram stands for 'good'. The diagram also shows the specifications of the pulse durations for the clock and data signals, as well as the minimum delay between the edges of the clock signal (level transitions) and the sampling point.

A swipe reader made by Hopt + Schuler was used for this project. It is distinguished by a wide operating temperature range (–20 °C to +60 °C) and a robust mechanical construction. The manufacturer specifies a useful life of 300,000 operations for the reader and 1000 operations for an individual magnetic-strip card. The card feed speed may range from 10 to 1000 mm/s. The reader is available in 1-track, 2-track and 3-track versions, with the track locations corresponding to the ISO standard (see the text box). The reader has a crimp-contact plug for connecting to the evaluation logic (in our case, the interface). This connector has 5 pins for the 1-track version, 9 pins for the 2-track version and 12 pins for the 3-track version. The interface circuit board has connector strips for all three versions. However, only the 1-track version was used in our prototype for testing, since only this type is available as a one-off (from Conrad Electronics).

Track adjustment

The 1-track reader is set up for the ISO-3 track. However, the manufac-

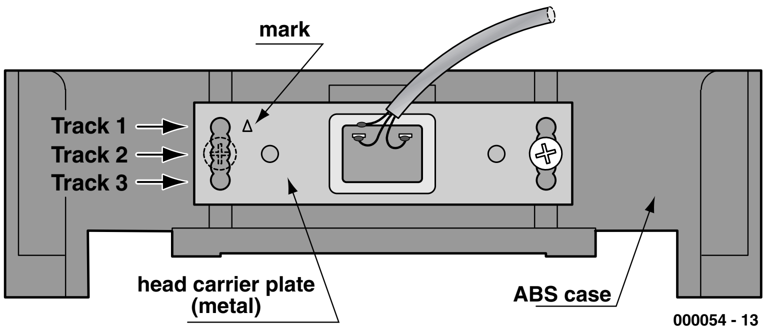


Figure 2. With the 1-track reader, the track position can be changed by repositioning the read head carrier plate.

Table 1

Jumper settings

1-track card reader: jumper on K2

Track	installed jumper
1	1–2 and 7–8
2	3–4 and 9–10
3	5–6 and 11–12

2-track card reader: jumper on K3

Track	installed jumper
1	1–2 and 5–6
2	3–4 and 7–8

No jumpers are necessary for a 3-track card reader.

turer has made provisions for adjusting the position of the head carrier plate, and thus the track position. This is shown schematically in **Figure 2**. The reader comes from the factory with the head in the highest position, as seen from the base plate. This corresponds to the ISO-3 track. The carrier plate can be set to five different positions using the two screws. Only the top, middle and bottom positions are relevant with regard to the ISO track locations. The top position corresponds to ISO track 3, the middle position to ISO track 2 and the bottom location to ISO track 1. If you always want to read only one particular track, then you only have to adjust the head position once. If you frequently need to read different tracks, this sort of repositioning is too inconvenient, since the enclosure must be opened and closed each time. However, there is a practical solution to this problem. You can simply leave the head in the top position, as delivered, and alter the position of the base plate instead. You can do this by placing one or two strips, each approximately 4 mm thick, in the bottom of the slot. These ‘underlays’ raise the card posi-

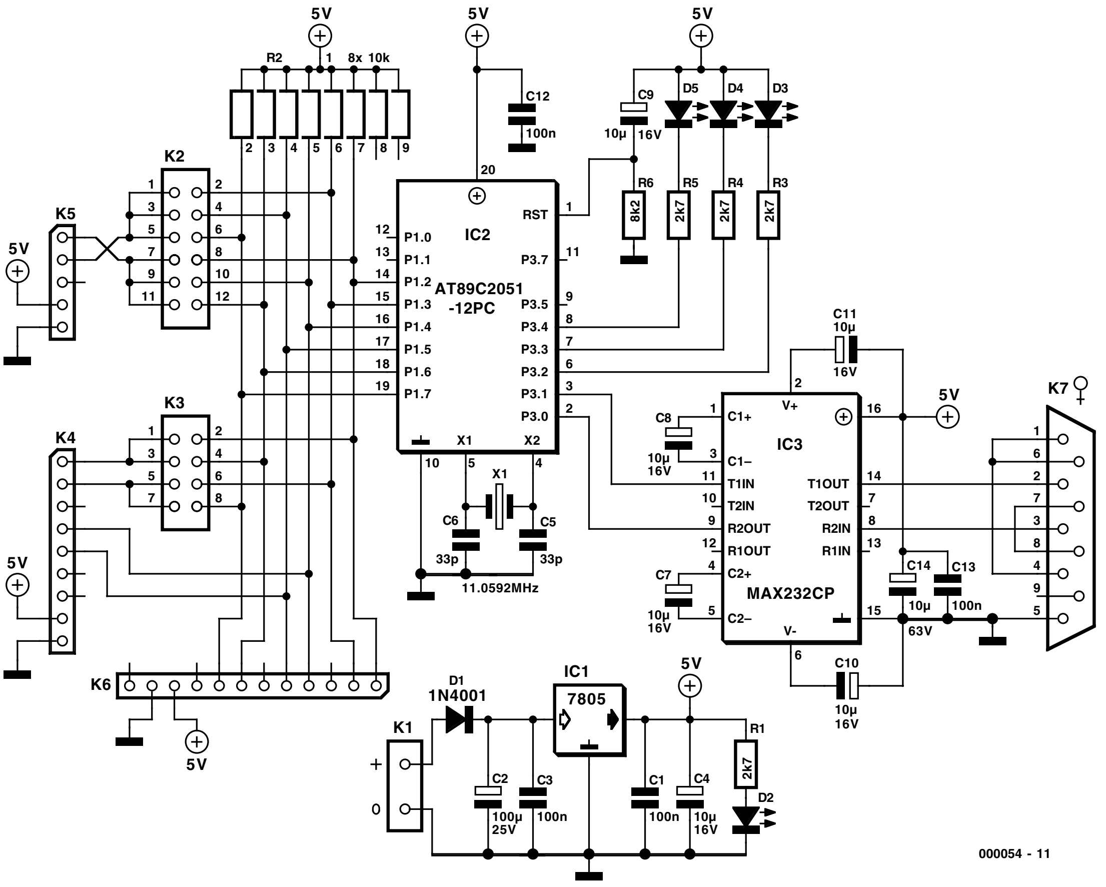


Figure 3. Circuit diagram of the interface circuit that connects the card reader to the serial port of a PC.

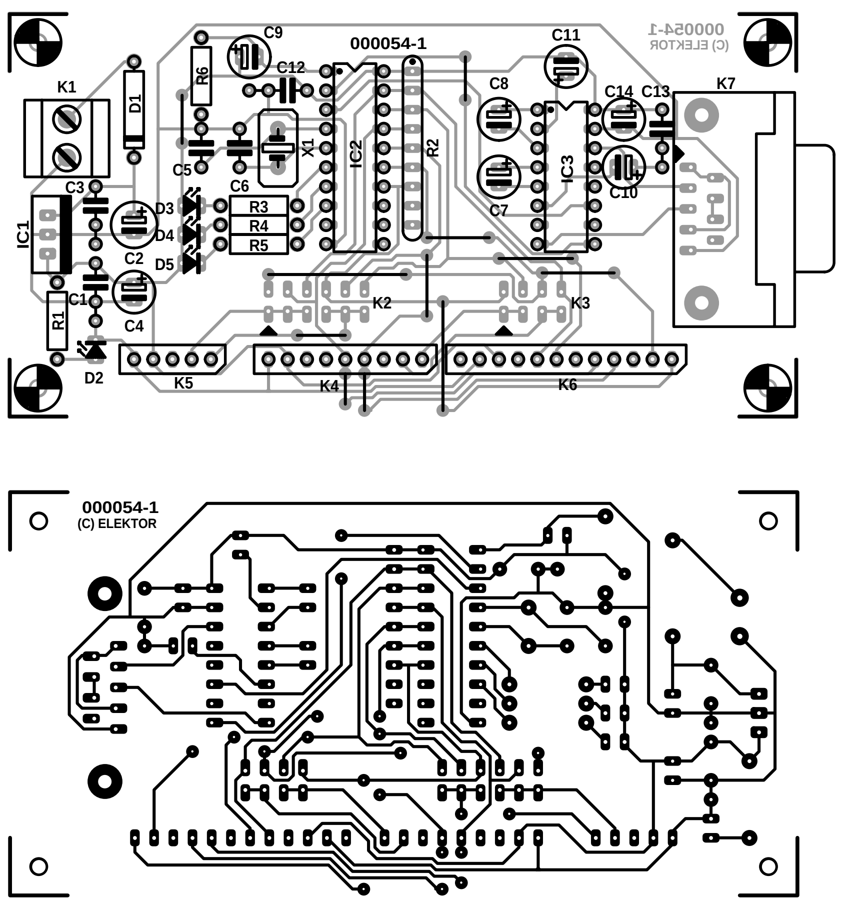


Figure 4. Layout of the single-sided printed circuit board

COMPONENTS LIST

Resistors:
R1,R3,R4,R5 = 2kΩ7
R2 = 8 x 10kΩ
R6 = 8kΩ2

Capacitors:
C1,C3,C12,C13 = 100nF
C2 = 100µF 25V radial
C4,C7-C11,C14 = 10µF 16V radial
C5,C6 = 33pF

Semiconductors:
D1 = 1N4001
D2,D3 = LED, low current, red, 3mm
D4 = LED, low current, yellow, 3mm
D5 = LED, low current, green, 3mm
IC1 = 7805
IC2 = AT89C2051-12PC (programmed, order code **000054-41**)
IC3 = MAX232CP

Miscellaneous:
K1 = 2-way PCB terminal block, raster 5mm
X1 = 11.0592 MHz crystal
K2 = 6-way Jumper
K3 = 4-way Jumper
K4 = 9-way pinheader
K5 = 5-way pinheader
K6 = 12-way pinheader
K7 = 9-way sub-D socket (female), angled, PCB mount
PCB, order code **000054-1**
Magnetic card reader e.g., Hopt & Schuler Type 832-01320000000 (order code 165328-33), see text
Disk, order code **000054-11**

tion when it is pulled through by either 4 mm (for track 2) or 8 mm (for track 1). This allows the reader to be set up for all three tracks without actually modifying the reader.

Interface

The schematic diagram of the interface is shown in **Figure 3**. Apart from the voltage regulator, it consists of only two ICs: a micro-controller and a MAX232. The PIC converts the clock and data signals from the card reader into a serial data signal for the PC, and it also controls communications over the serial interface. The MAX232, as usual, converts signal levels between TTL and RS232 in both directions.

The three separate connector strips can be seen in the schematic diagram and on the printed circuit board. As already mentioned, these are intended to be used for the three different versions of the Hopt & Schuler card reader. The 5-pin connector K5 is intended to be used with the 1-track reader described above. The circuit can be configured for the version of the card reader that is actually used via jumpers placed on the pin strips K2 and K3, as indicated in **Table 1**.

There is nothing particularly unusual about the construction of the single-sided printed circuit board shown in **Figure 4**. It hardly needs to be said that you should inspect the board after assembly to be sure that all wire jumpers are in place, all diodes and electrolytic capacitors have been fitted with the correct polarisation and the ICs are soldered in (or inserted in the sockets) the right way around. One important point is the use of a sub-D socket for K7. In the experience of the *Elektor Electronics* design staff, the most common cause of problems with the connection between a Elektor project circuit and PC serial port is that the builder has used a sub-D plug on the circuit board instead of a sub-D *socket*, and thus used the wrong cable.

An unstabilised 9-V mains adapter that can supply around 100 mA is an adequate source of power for the interface circuit.

On Project Disk #000054-11

MAGREAD	EXE	Windows program
MAGREAD	HEX	hex file
MAGREAD	ASM	assembler file
INFO	TXT	text file
COPYRIGHT	TXT	text file
CONTENTS	TXT	text file

Operation

When the circuit is switched on, the pilot lamp LED D2 illuminates. The three track-indicator LEDs (D3, D4 and D5) are alternatingly illuminated

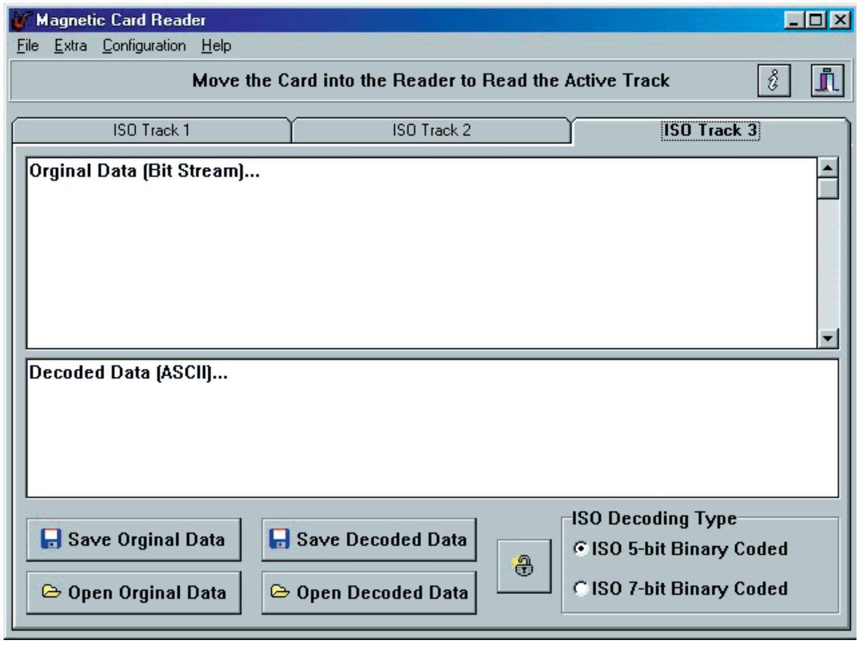


Figure 5. User interface of the decoder program.

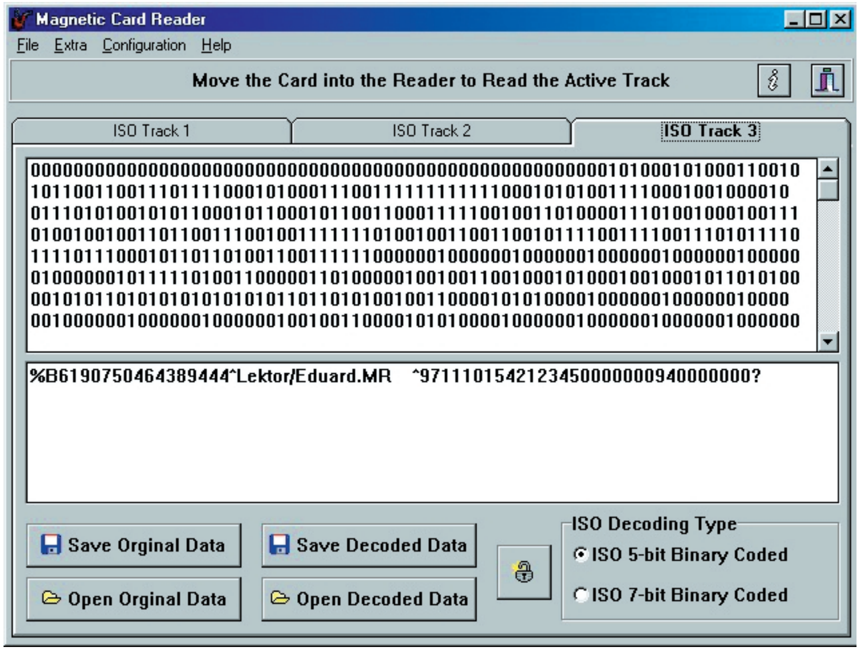
as a running-light display during the following initialization phase. On completion of the initialization, only D5 remains on, which shows that track 3 has been selected. As shown in **Figure 5**, the Windows PC program Magread.exe also starts up with track 3 as the default selection. You can select any one of the three tracks on the card by switching the tab cards 'ISO Track 1', 'ISO Track 2' and 'ISO Track 3'. The indicator LEDs on the interface card will change to match the selected track. When a new track is selected, the word format is automatically changed to the value assigned by ISO (5 bits or 7 bits). However, if you wish you can switch between 5-bit and 7-bit coding for any track.

The rest of the operation of the program is very simple. Two entries under the 'Extra' menu provide for the convenient display of data from a Eurocheque or bank card and a credit card. This is done as follows. With a Eurocheque or bank card, first read ISO track 3. You can then press function key F5 to display the card data (bank sorting code, account number, expiry data and so on) in a separate window. With a credit card, read ISO track 1 first, and then press function key F6 to display the card data (card number, cardholder name and expiry date) in a separate window.

Using the Configuration menu, you can select the Com port (Com Port Setup), set the decoding delay (Decode Delay) and select the language for using the program (Deutsch or English). The decoding delay sets the time between the start of reading and the start of decoding, which depends on the type of PC used. The standard setting is 2500 ms (2.5 s), which is valid for 486/33 MHz PCs and upwards. If the data are not correctly decoded (as indicated by an error message), increase the decoding delay.

The program shows the original data that is received as a bit stream from the card reader at the top of the control window, and the decoded data (in ASCII) at the bottom of the window. You can also start the decoding by clicking on the padlock icon at the bottom of the control window. **Figure 6** shows how the data from a credit card are displayed (with fictitious data for privacy protection). Starting at the left, the first two characters are start characters. These are followed by the 16-digit card number, followed by a '^' as a separator. Next come the last name and first name of the cardholder, and then a period and the title. The four characters after the next separator indicate the expiry date of the card (in this case '9711', which means 11/97). The following characters represent an encrypted code for the credit card firm, which allows an offline credit card reader to verify the card number.

The control boxes at the lower left (Save and Open for original and decoded data) allow you to save and restore the displayed information.



```
0000000000000000000000000000000000000000000000000000000101000101000110010
10110011001110111100010100011100111111111111000101010011110001001000010
011101010010101100010110001011001100011111001001101000011101001000100111
010010010011011001110010011111110100100110011001011110011110011101011110
111101110001011011010011001111111000000010000000100000001000000010000000100000
01000000010111110100110000011010000001001001100100010100010010001011010100
0010101101010101010101010110110101001001100001010100001000000010000000100000
001000000010000000100000001001001100001010100001000000010000000100000001000000
```

```
%B6190750464389444^Lektor/Eduard.MR   ^97111015421234500000000940000000?
```

Figure 6. Sample decoded data for a fictitious credit card.

Conclusion

As already noted, the interface circuit performs a self-test after power is applied. This is indicated visually by running-light display using the the ISO track selection indicator LEDs. This should move back and forth one time. If this is not the case, check the circuit for faults (defective components or construction errors). When the circuit is working properly, it sends a startup message to the PC via the serial interface when power is applied.

Here are some troubleshooting tips:
Fault: no magnetic card data appear

1. Verify the COM port settings
2. Check for faults in the interface cable.
3. Check the circuit board for faults.

Fault: only corrupted or erroneous data are displayed on the PC

1. The card was not properly pulled through the reader.
2. The PC (interface card) cannot handle 57,600 baud. Use a different interface card (with a 16550 UART) or a different PC.

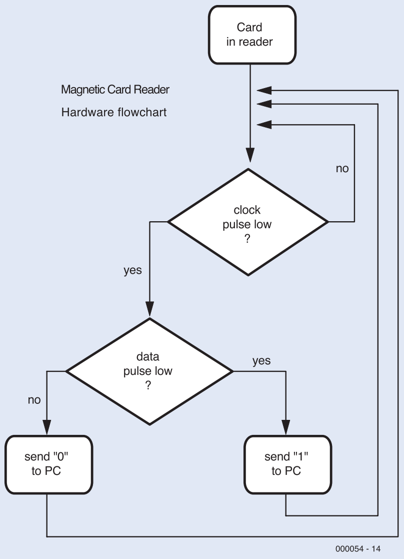
Fault: error message 'Decoding Error!, Start Sentinel not Found!' is displayed

1. The card was not properly pulled through the reader.
2. The card was pulled through the reader in the wrong direction (pay attention to the arrow on the reader).
3. The time between the start of reading and the start of decoding is too short. Change the decoding delay (Configuration menu) from the standard 2500 ms to a larger value.
4. The startup message from the circuit is transferred to the PC.

(000054-1)

Internet addresses:
Information on magnetic-strip cards:
www.paulmax.eng.net./indexmag.html
Swipe readers are available from the manufacturer (*www.Hopt-Schuler.com*) and from *www.conrad.de.*

Software



The magnetic-strip card reader reads and decodes data as follows. The program in the hardware circuit waits in an infinite loop until a card is fed into the card reader. As soon as the clock output of the reader module, which has negative logic, changes to a Low level, the program in the Atmel AT89C2051 microcontroller tests whether the signal at the data output of the reader has a High level (corresponding to a logical '0') or a Low level (corresponding to a logical '1'). According to the level that it finds, it sends a '0' or a '1' via the serial interface, which runs at 57.600 baud with 1 stop bit and no parity.
After the microcontroller program has sent a '0' or a '1' via the serial interface, it waits for the next Low level of the clock signal.
The data are output in 'clear' text, which means that they can be displayed using any terminal emulation program with the above-mentioned interface settings. The data stream could for example appear as follows in the terminal program:

```
000011010000110000010001100100100101010110111100000101001111111
```

(This can be evaluated as 'S0123456789E', where 'S' = start character and 'E' = end character). These are the 'raw' data that are transferred from the interface circuit to the PC. They are evaluated by the program running in the PC. Since the interface sends only raw data to the PC, the data can be used in programs that you develop yourself, and the equipment can also be used for special magnetic card codings that are not ISO compliant.
The software in the interface reads in the data string when the card is pulled through the reader. The decoding software starts evaluating the raw data after a software-adjustable time delay, which the software needs to allow the serial buffer to be completely read in and emptied. The decoding software can work with ISO 5-bit and ISO 7-bit encoded magnetic-strip card data.
The data are decoded in the following manner. A software routine searches for the start character in the raw data string. This is the character string '11010' for ISO 5-bit data, or '1010011' for ISO 7-bit data. Once this is found, the next five or seven characters are read in and decoded by a software routine. The following is an extract from the source listing for ISO 5-bit data decoding:

```
function ISODecode_5bit_Coded(InputStr: String): String;
begin
   ISODecode_5bit_Coded := '?';
   if InputStr = '00001' then ISODecode_5bit_Coded := '0';
   if InputStr = '10000' then ISODecode_5bit_Coded := '1';
   .......
   if InputStr = '10011' then ISODecode_5bit_Coded := '9';
   if InputStr = '11010' then ISODecode_5bit_Coded := 'S'; {start character}
   if InputStr = '00111' then ISODecode_5bit_Coded := '<'; {control}
   if InputStr = '10110' then ISODecode_5bit_Coded := '='; {control character}
   if InputStr = '01110' then ISODecode_5bit_Coded := '>'; {control}
   if InputStr = '11111' then ISODecode_5bit_Coded := 'E'; {end character}
end;
```

Since according to the ISO standard, only numeric and control characters are present on tracks 2 and 3, 5-bit decoding is sufficient for these tracks. Track 1 contains numeric, alphanumeric and control characters, so 7-bit decoding is necessary.

Valve Preamplifier (I)

Based on the ECL86

Design by G. Haas – Email: experience.electronics@t-online.de

The good old valve amplifier is experiencing a renaissance. Consequently, we present a valve preamplifier that unquestionably belongs to the high end.

In the high-end region, valve amplifiers enjoy uninterrupted popularity. Although a lot of things can be done with modern semiconductors, the old-fashioned valve is still very much in the news in elevated audio circles. In both domestic and studio equipment, valves are being used increasingly often in equipment such as compressors, equalisers, simple amplifiers, filters and the like. The objective is to achieve a warmer, more attractive sound than what can be obtained using only sterile semiconductor technology. The digital era in particular gives many recordings an unpleasant sharpness, which can be moderated by the knowledgeable and consistent use of valve technology. The valve preamplifier described here usually makes CDs sound more pleasant than they other-

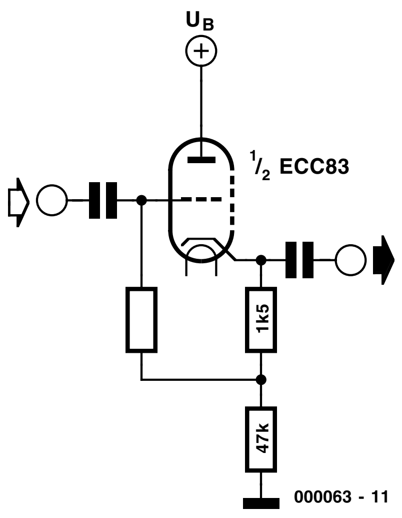


Figure 1. A standard valve preamplifier configuration using one ECC83 for the two channels.

wise would.

This preamplifier is a no-compromise design. Only valves are used in the signal path, while semiconductors are used for the auxiliary functions. In this way, the two technologies complement each other. To be consistent, we have also avoided using semiconductors for switching in the signal path.

The valve determines the design

The choice of the amplification elements, the valves, largely determines the topology of the circuit. Given the limited selection of suitable valves, a few basic designs have come to predominate. However, these all have certain significant disadvantages.

A typical preamplifier, for example, is fitted with ECC81, ECC82, ECC83, ECC88 or similar valves. The ECC83 has a high no-load gain, but only a small working current (1 to 1.5 mA). The ECC81 and ECC82 have lower gain, but they can be operated at currents up to 10 mA. The ECC88 (or the equivalent type PCC88) is most commonly used in television sets, but it is also popular for low-frequency applications, since it can work at currents of up to 15 mA, with an operating voltage of only 90 V.

Connecting two double triodes in series, however, provides far more gain than is needed nowadays, while still not providing an adequate amount of current. In order to reduce the output resistance, a cathode follower circuit is frequently used. This significantly diminishes the dynamic output resistance, although it does not completely eliminate it. A typical cathode follower circuit using an ECC83 is shown in **Figure 1**. The dynamic output resistance R_d is given by the formula:

$$R_d = \frac{R_i \cdot R_k}{R_i + R_k \cdot (\mu + 1)}$$

With the following typical values for the ECC83, this yields the following value for R_d:

no-load gain:	$\mu = 100$
internal resistance:	$R_i = 62.5\ \text{k}\Omega$
anode current:	$I_a = 1\ \text{mA}$
cathode resistance:	$R_k = (47 + 1.5)\ \text{k}\Omega$

$$R_d = \frac{64.5 \cdot 48.5}{62.5 + 48.5 \cdot (100 + 1)}$$

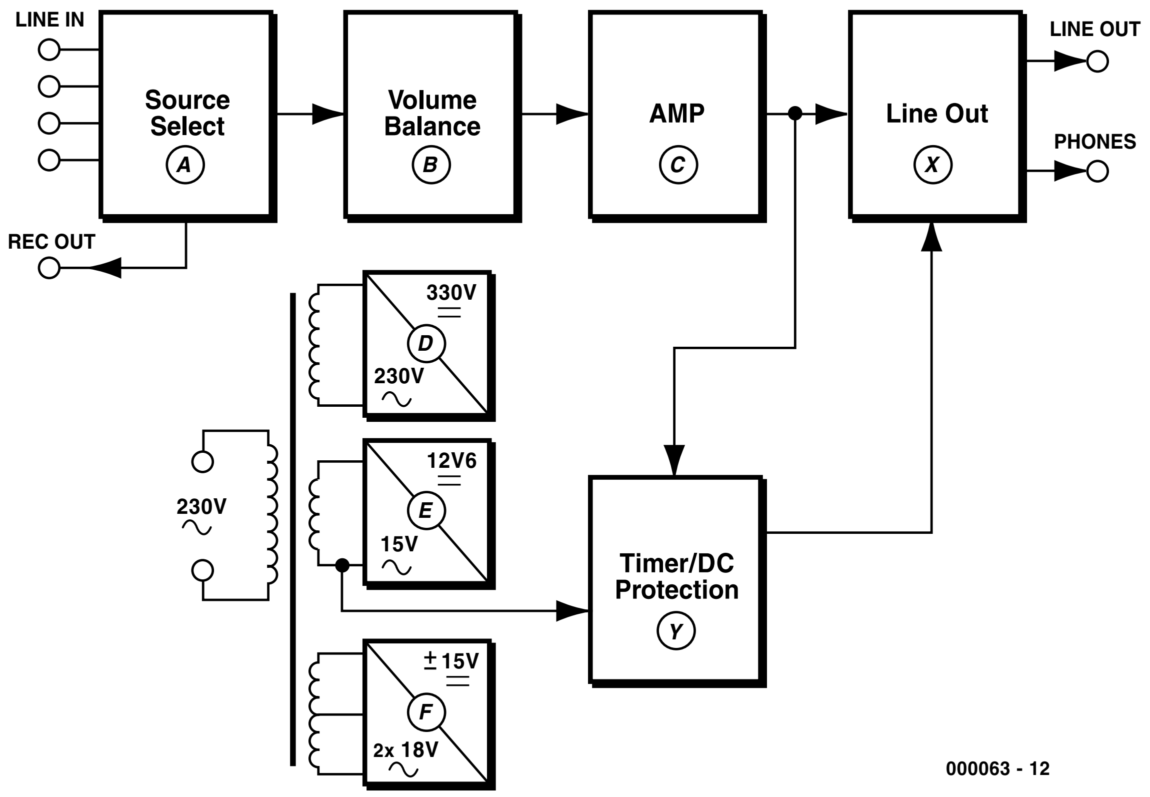


Figure 2. Block diagram of the preamplifier, showing its modular design.

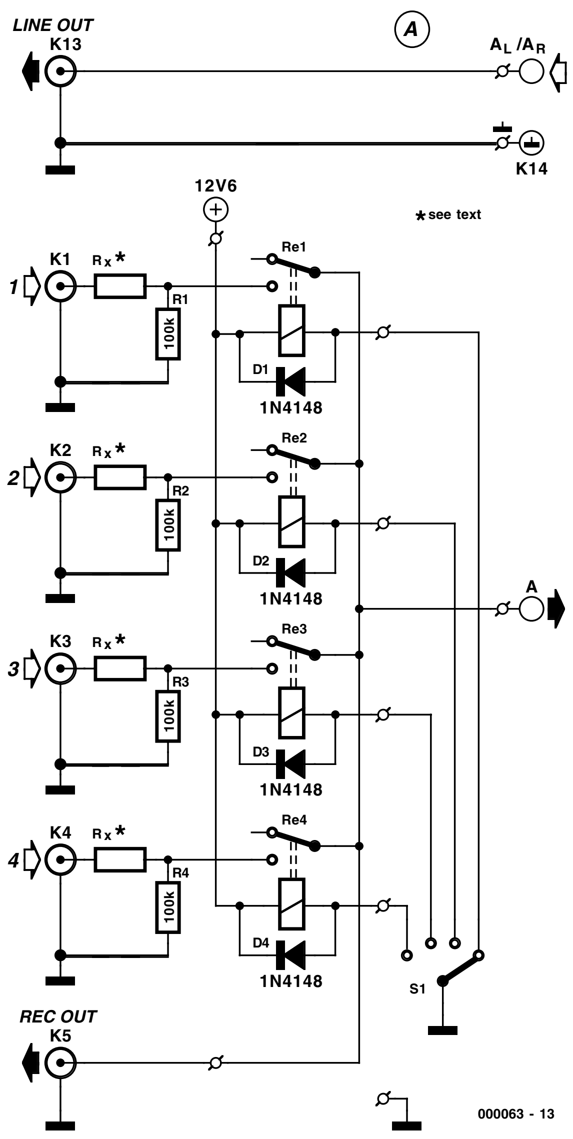


Figure 3. The external connections: the schematic diagram of the relay board with the inputs and outputs.

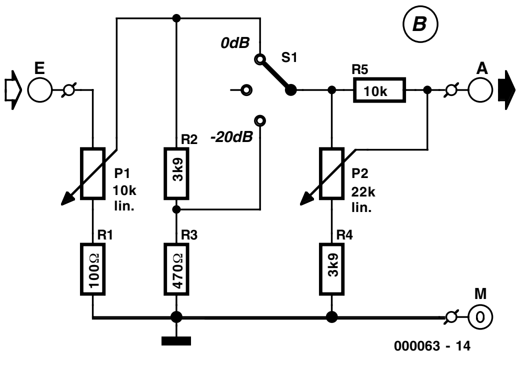


Figure 4. Here is where the volume and balance are adjusted.

This appears to yield a low output impedance. However, in actual fact the total cathode resistance (48.5 kΩ) or the internal resistance of the valve will determine the effective output resistance of this arrangement (strictly speaking, only if it is overdriven). If we assume that the net capacitance of the circuit and the output cable is only 500 pF, which is easily possible with cables that are a few metres long, the attenuation at 20 kHz is 14 dB. This is why you often see recommendations to limit the cable length to 1.5 metres and to use the lowest-capacitance cable that is available. This certainly helps to reduce the problem, but it does not eliminate its source. The sound of the system will be audibly different when different types of cable are used, and this is a natural consequence of the construction of the amplifier.

Using a single valve for both stereo channels is a mistake that has been inherited from the early days of stereo hi-fi technology, and which has proven to be almost impossible to eradicate. The channel separation suffers, due to capacitive crosstalk within the valve and in wiring of the valve socket, and this considerably impairs the spaciousness and detail resolution of the sound.

Valve amplifiers are often operated without negative feedback. This may not have caused any problems in the days of monophonic sound, but the only way to guarantee proper stereo reproduction is to use strong feedback to achieve equal performance in the two channels of a stereo system, regardless of tolerance variations in the characteristics of the valves. This also causes the distortion factor to remain very low, even when the amplifier is driven hard, and it flattens the frequency response. This also satisfies the demand for the lowest possible distortion in the preamplifier.

For all of these reasons, the preamplifier described here does not follow the conventional path. An ideal amplifier has a high input impedance, a high no-load gain and a low output impedance. These conditions are easily satisfied by operational amplifiers using semiconductor technology. With valves, the situation is far more difficult. In order to avoid the above-mentioned design shortcomings, an ECL86 double valve is used here. The triode section of this valve is exactly the same as that of an ECC83. The pentode section can be used as a power amplifier that can deliver 4 watts with an anode current of 36 mA and a distortion factor of 10 percent. If we properly combine the triode and pentode sections, we can obtain a sort of valve operational amplifier with good characteristics that are similar to those of a modern semiconductor opamp.

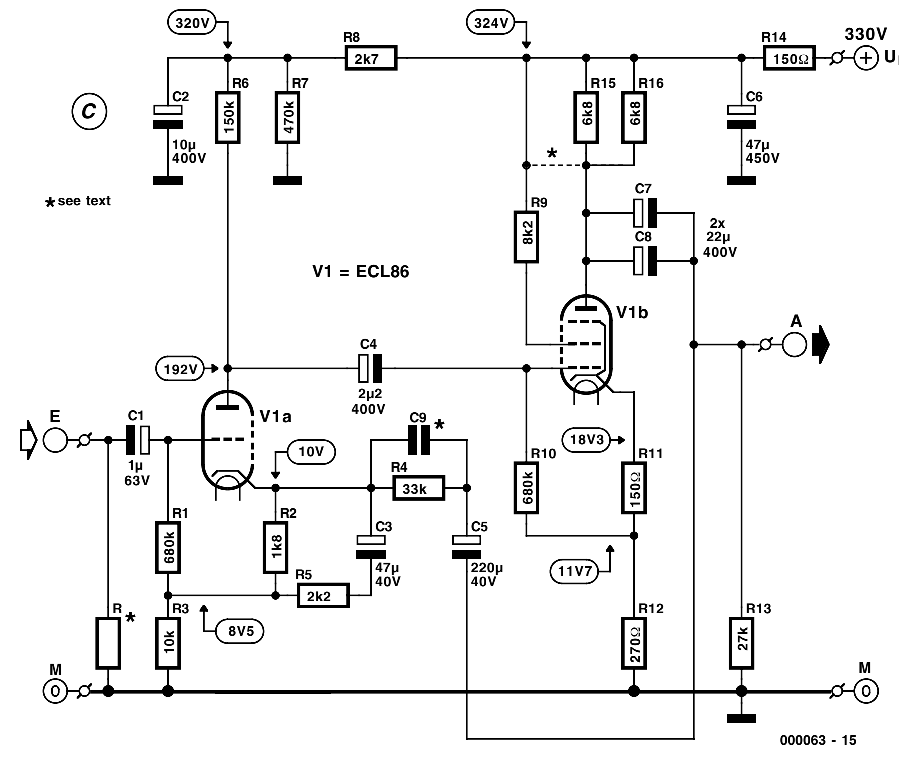


Figure 5. The highlight of the design is the valve amplifier stage.

Modular design

The block diagram of one channel of the preamplifier is shown in **Figure 2**. A modular approach is used in the design, with each functional group located on a separate printed circuit board. The amplification chain consists of four elements, which are the input selector circuit, the volume/balance adjustments, the actual preamplifier and a changeover switch for line or headphone outputs. A protective circuit with a time delay switches the outputs to earth when a fault is detected in the output signal. An external power supply is used. The modular construction not only allows the circuit to be modified relatively easily, it also promises very good values for channel separation and the signal-to-noise ratio. This makes up for the increased cost and effort for the wiring.

In the following descriptions of the individual modules, the component numbers correspond to the labels on the circuit boards. In contrast to the usual practice with Elector projects, the components are not numbered sequentially for the complete circuit, but instead separately for each module. Only one channel of the amplification chain is shown; the component numbers for the second channel are either shown in parentheses or are distinguished by a 'prime' mark (').

Input selection with relays

The input selection module shown in **Figure 3** is laid out for four signal sources K1 through K4 (K6 through K9) that are connected to a single common line via the single-in-line reed relays Re1 through Re4 (Re5 through Re8). The consistent use of screening and the split layout of the printed circuit board provide a high degree of channel separation from the very start.

The 100-kΩ resistors R1 through R4 (R5 through R8) terminate the input sockets and conduct any static charges to earth. They prevent clicks when a different input source is selected or the input cables are reconnected. The resistors labelled Rx can be used with signal sources that have different output levels; their values can be chosen as necessary. However, you should make sure that the series resistance for K5 (K7) matches the source resistance for the tape recorder output. If the resistance is too large, high tones will be lost.

The 12-V relays, which are powered by the +12.6-V DC filament supply voltage, are switched by the

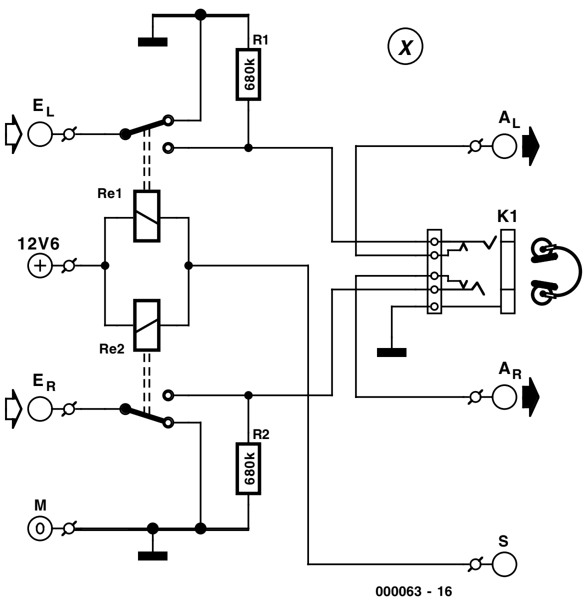


Figure 6. The headphone and line output connections.

rotary switch S1. This is connected to the circuit board via solder posts.

The line out sockets K13 (K15) and the earth terminals K14 (K16) are located on the same circuit board, but they are fully electrically isolated from the rest of the board. The tape recorder output sockets must be mounted externally, and then connected to solder posts K5 and K7 (earth to K17).

Volume adjustment

From connector K5 (K7) of the relay board, the signal is conducted to the circuit shown in **Figure 4** for adjusting the volume, balance and muting. The potentiometers that are used, with 41 detent positions, have good reproducibility. The balance adjustment also has a fixed midpoint setting. The mute switch S1 spreads the adjustment range, so that even low volume levels can be sensitively adjusted. If you find the 20 dB step unsuitable, you can modify the values of the resistors. The sum of R2 (3.9 kΩ) and Re (470 Ω) should remain approximately the same, but the individual values can be modified as desired. When the mute switch is in the middle position, the signal path is interrupted, and the output is muted.

The characteristic curve of the linear volume adjustment potentiometer P1 (which has good tracking due to its linearity) is slightly modified by inserting R1 between its lower terminal and earth. This compensates for the slight imbalance between the two channels in the initial region of the volume adjustment, which arises from the construction of the preamplifier. Of course, with such an arrangement it is not possible to set the volume level fully to zero.

The balance control P2, in combination with R4 and R5, has an adjustment range from around +3 dB to –4 dB. This is fully adequate for correcting the midpoint. With the indicated component values and the 100-kΩ resistors on the relay board, at the middle position of the balance control the input resistance is approximately 2.2 kΩ. This is a suitable value for all modern signal sources, and it is low enough to keep the disturbance sensitivity and the total noise at low levels.

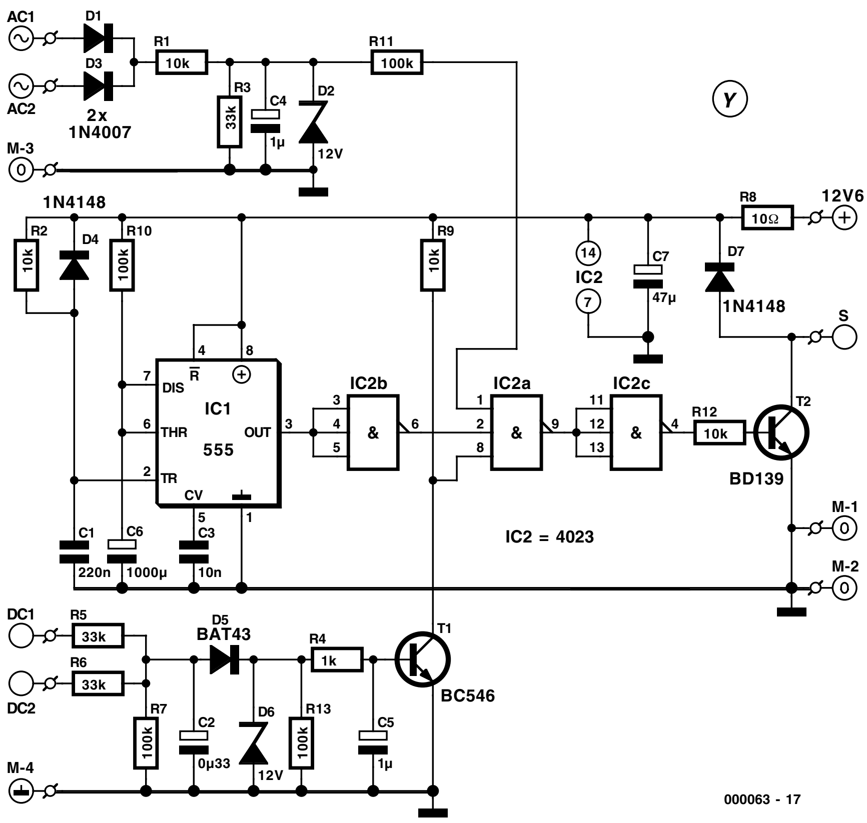


Figure 7. The protective circuitry monitors the state of the preamplifier and switches the output relays.

The amplifier

The amplifier module shown in **Figure 5** makes a familiar impression. Valve V1a is the triode section of the ECL86, equivalent to half of an ECC83. The DC operating point is set to just below 1 mA by R2. The amplified signal reaches the pentode section V1b via capacitor C4. Neither section of the valve is wired in the conventional manner. The cathode and the grid bleeder resistor are not connected directly to earth, as is usual, but instead via resistors R12 and R3, respectively. This provides local negative feedback for each valve section. This constrains the valves to work at well-defined operating points, so it is not necessary to use selected valves.

The load resistance of V1b consists of R15 and R16 in parallel (each 6.8 kΩ, 4.5 W). This divides the not insignificant power dissipation equally over two component packages. The signal is coupled out via two parallel 22-μF capacitors (C7 and C8). Using two capacitors instead of a single 47-μF capacitor halves the ESR (effective series-resonant resistance) of the output capacitor, which benefits the high frequency transfer characteristic of the amplifier. Resistor R13 bleeds off static charges.

To ensure that the circuit has a well-defined behaviour, the output signal is coupled back via C5 and R4 in negative feedback. The basic amplification of V1a depends on the load resistance of R6, the operating voltage (the voltage across C2) and the cathode circuitry. Resistors R2 and R5 are connected in parallel for AC signals. The amplification is thus determined by the relationship of the parallel combination of R2 and R5 to the resistance of R4. If you want to modify the basic amplification, you can change the values of R5 or R4. The smaller the resistance of R4, or the larger the resistance of R5, the higher the gain. Capacitor C5 separates the AC and DC currents, so that the DC operating point of V1a does not shift. Resistor R2 may not be modified to change the gain, since this would shift the DC operating point in an unacceptable manner.

The supply voltage, U_B, is 330 V. It is decoupled and filtered by R14 and C6, in order to avoid channel crosstalk via the supply voltage. The supply voltage is further decou-

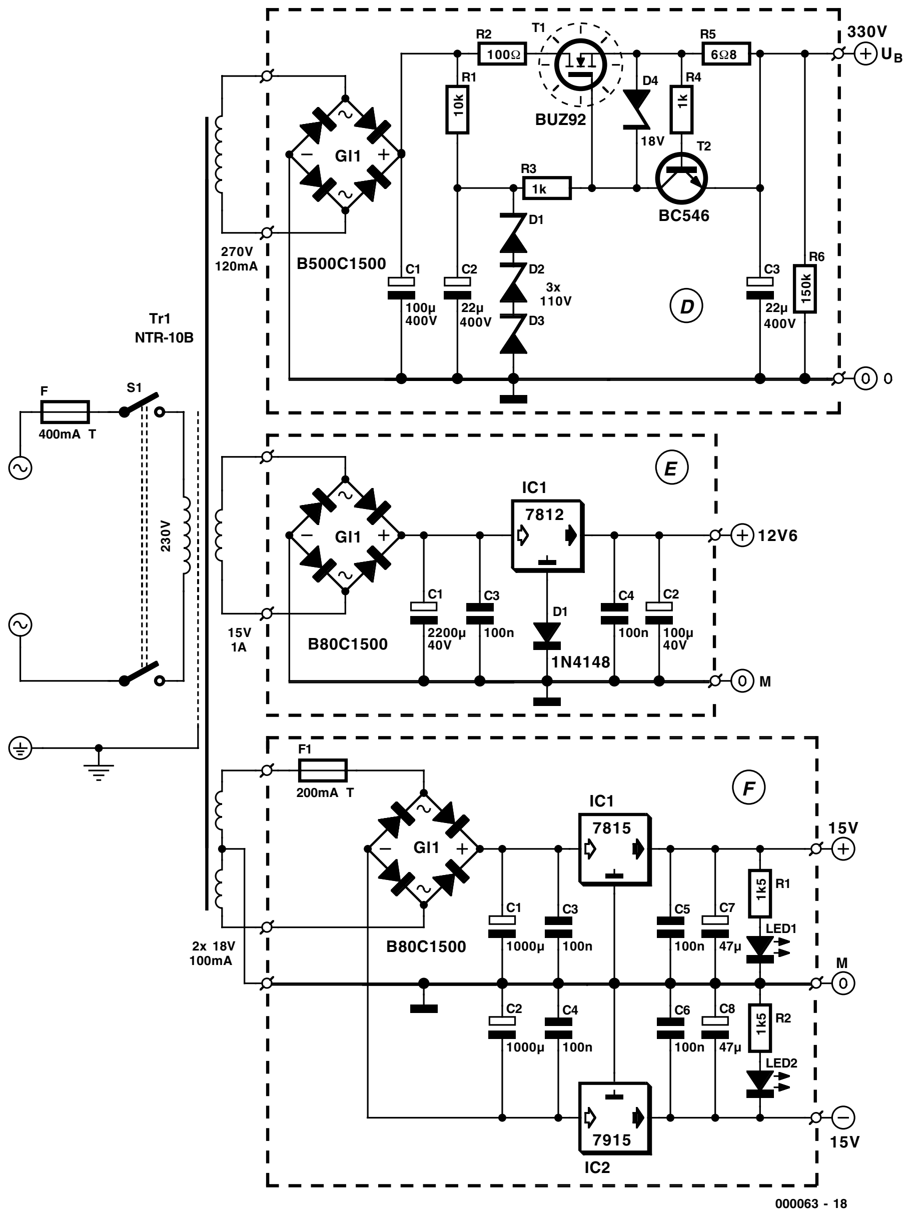


Figure 8. The power supply is a combination of a high voltage supply, a low voltage supply and an optional third supply.

pled and filtered for V1a by R8 and C2. Resistor R7 is a bleeder resistor that discharges the high-voltage electrolytic capacitors after the supply voltage has been switched off.

Options

The input resistor R can be omitted, since the volume and balance adjustments also shunt static charges. However, if the amplifier module is used in some other application, any desired value can be used as necessary. The indicated connection between R9 and the anode of R1b should only be used if you want to operate the pentode in quasi-triode mode. In this case the combination of R15 and R16 is shorted out, and R9 is naturally redundant.

Line or headphones

Using the ECL86 as a preamplifier valve has the additional advantage that the pentode section can deliver 4 watts of output power. This capability can be used to advantage by connecting a switched phone socket (K1) between the amplifier output and the line output. When a headphones plug is inserted into this socket, it automatically disables the line outputs. The pentode section of the ECL86, which can deliver a relatively large current, can easily drive headphones with an impedance greater than 300 Ω, even over a relatively long screened cable, without degradation of the sound quality. Normal screened cables are fully adequate for this purpose.

The portion of the circuit shown in **Figure 6** also includes two relays, which are driven by the protective circuitry. The two resistors bleed static charges to earth.

Unconditional protection

The protective circuitry shown in **Figure 7** performs several tasks at the same time. Three conditions must be satisfied before the preamplifier is switched 'online'.

When the amplifier is first switched on, the outputs are initially shorted out via the relay outputs. The timer output of IC1 (pin 3) goes Low after approximately 100 s, at which time the valves should be properly warmed up and all charging processes completed. This delay prevents humming, burbling or other upsetting sounds from being generated by the speakers while the preamplifier is heating up. The output of inverter IC2b then sets input 2 of NAND gate IC2a High.

If a voltage is then present at pins AC1 and AC2, which are connected directly to the filament winding – diodes D1 and D3 monitor both half cycles of the AC waveform – then C1 is charged via R1 to 12 V (limited by D2). A High level is thus applied to input 1 of IC2a.

The DC components of the two output signals are monitored via connections DC1 and DC2. A DC voltage can for example be present if an output coupling capacitor breaks down. The AC components are shorted out by C2, which is discharged by R7. This combination also determines the time constant of the monitoring circuit. If a DC voltage higher than around 1.3 V is present, transistor T1 is driven via D5 and R4. This pulls input 8 of gate IC2a Low. A Schottky diode with a very low forward voltage drop is used to ensure that the monitor circuit will respond, even with very low values of the DC component. Transistor T1 is cut off only when no significant DC component is present, so that R9 can pull input 8 of gate IC2a high. Resistor R13 ensures that T1 is securely cut off in this situation. Diode D6 limits the maximum voltage applied to the base of T1, while R4 and C5 provide a short time delay, so that the circuit does not immediately respond to every tiny disturbance.

Only when all three of these conditions have been satisfied, so that all three inputs of the AND gate are High, will the protective circuit enable the output relays via inverter IC2c and the driver transistor T2.

When the mains transformer is switched off, the level at input 1 of the gate goes Low almost immediately. This causes the relays to be immediately disabled, so that no disturbance signals can be passed on to the following equipment while the capacitors in the preamplifier are discharging. The reset circuit consisting of R2, D4 and C1 also resets timer IC1 after a brief interruption of the mains power, so that the full delay time must expire each time. The delay time can be changed by modifying the values of R10 and C6. The time is equal to 1.1·RC. The protective circuitry is operated from the filament voltage power supply, with a voltage of 12.6 V.

One transformer, three power supplies

A good power supply is essential for the proper operation of a preamplifier. Since the amplifier draws relatively little current, a cleanly filtered and stabilised power supply can be built economically using modern semiconductors. **Figure 8** shows the schematic diagram of the entire power supply circuit. This is also built using several separate circuit boards, which are indicated by dashed outlines.

The preamplifier needs a high voltage for the valves, as well as a low DC voltage for the filaments, the relays and the protective circuitry. These voltages, as well as some others, are provided by a single mains transformer (NTR-10B) that is protected by a 0.4-A slow-blow fuse on the primary side. This transformer is made using grain-oriented 0.35-mm steel laminations with especially low dispersion and losses, which are usually used for high-quality low-frequency coupling transformers. A very orderly winding technique and vacuum impregnation, which is not possible with toroidal transformers, provide long-term stability and corrosion resistance. The impregnating resin penetrates into the coils and permanently anchors each individual winding. Electrical safety is provided by a 4000-V breakdown test between the primary and secondary windings and a static screen that is connected to the protective earth lead. Obviously, you will not find this sort of high-end transformer in every electronics supermarket. It is only available from Experience Electronics, the author's firm.

At the top of the schematic diagram, you can see the high voltage section. This circuit filters the hum voltage to a level that is below the intrinsic noise level. Resistor R1 and diodes D1 through D3 generate a good reference voltage with a value of around 330 V. Series regulator transistor T1 is a high-power V-FET (type BUZ92) in a TO-220 package. This allows the high voltage power supply to be constructed in a compact manner, using a small heat sink. The heat sink is mounted on the printed circuit board, with makes for short leads in the high voltage path. Resistors R4 and R5 and transistor T2 provide current limiting. With the indicated component values, this is set to around 90 mA. Resistor R6 discharges the electrolytic capacitors after the power is switched off. Zener diode D4 limits the gate voltage of T1, which must not exceed 20 V.

In order to avoid hum voltages, the filaments are powered by a DC supply. The frequently heard assertion that using a DC current for the filaments damages the valves is simply nonsense. All that is necessary is to bring the filaments to a particular temperature, so that the cathode can emit enough electrons. Whether this temperature is achieved using an AC current or a DC current does not matter. The 'odd' value of 6.3 V for the filament voltage, by the way, comes from the early days of valve technology, when the filaments were powered by four carbon-zinc batteries. Since fresh batteries have a terminal voltage of more than 1.5 V, a margin of 0.3 V was chosen to prevent the filaments of the (at that time) very expensive valves from burning out prematurely.

Each of the two ECL86 valve filaments draws 0.66 A at 6.3 V. To minimise the losses in the filament circuit, the two filaments are connected in series. A 12-V voltage regulator connected to earth via a 1N4148 forms a simple but good 12.6-V power supply for the filaments. The relays for the input selector are also powered by this supply. The aluminium mounting plate is used as a heat sink for the voltage regulator, which must be mounted using an insulator.

At the bottom of the circuit diagram you can see a third power supply that provides symmetrical ±15 V. This supply is not necessary, so it is optional. It can be used to power external devices, such as an equalisation preamplifier, from a non-earthed supply. If such a supply is already present in the preamplifier, you save the cost of a mains adapter and the associated cabling, and the external equipment can be switched on and off at the same time as the preamplifier. The aluminium mounting bracket is also used as a heat sink for this supply, which must be isolated. The supply voltage has additional protection in the form of two fuses in the transformer leads.

(000063-1)

The second part of this article will include the printed circuit board and component layouts, the components list, precise construction information and, of course, performance data.

Tiny Webservers

Link your own project to the Internet

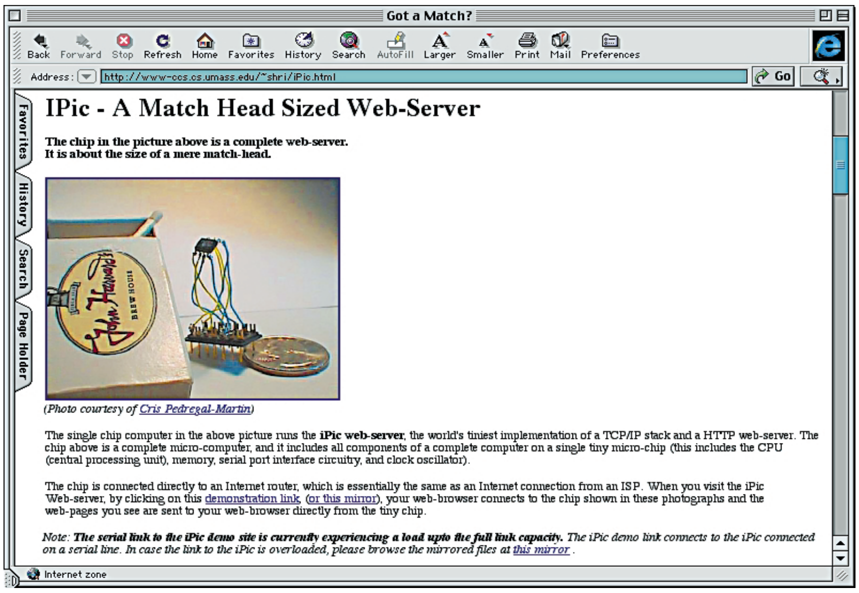
By G. Polder

E-mail, e-Commerce, and recently even e-Living, you mention it and it gets tied to the Internet. Lots of information is available on how to do that, as well as on the use of ever smaller webservers that come into play.

According to Scott McNealy, top executive of Java inventors Sun Microsystems, a car will soon be redefined as a virtual Java machine on wheels. According to Scott, it will not be long before household electronics not linked to the Internet will belong in a museum. After *e-commerce* and *e-business*, we now get *e-living*. The technology used by Sun for this purpose is called Jini, a network technology based on Java.

On *http://www.java.sun.com/jini* you may find lots of information on this promising technology. Many electronics manufacturers are busily working on Jini developments. Besides the Sun website, there is also an independent Jini site at *http://www.jini.org/*. The concept of Jini is very promising and although we are sure to see more of it in the near future, there is not a lot you can do with it as yet when it comes down to hands-on (hobby) electronics.

Fortunately that does not apply to the so-called *embedded web servers*. Although in principle Internet servers, these are stripped-down programs running on small processors like PICs. None the less, they do offer the basic functionality of Internet protocols like TCP/IP, allowing all manner of hardware to be coupled to the Internet. There are constant reports from different designers claiming to have designed to the smallest webserver. One the size of the head of a matchstick may be found at
http://www-css.cs.umass.edu/~shri/iPic.html
This server has been implemented on an 8-pin PIC type 12C509a. Denis Petrov on *http://www.chat.ru/~zhengxi/wwwpic/source.htm* shows a TCP/IP stack implementation and a www servers packed into a PIC16F84. The nice thing about this site is that the com-


Got a Match?
Back Forward Stop Refresh Home Favorites History Search AutoFill Larger Smaller Print Mail Preferences
Address: http://www-ccs.cs.umass.edu/~shri/iPic.html
Go
Favorites History Search Page Holder
Internet zone

IPic - A Match Head Sized Web-Server

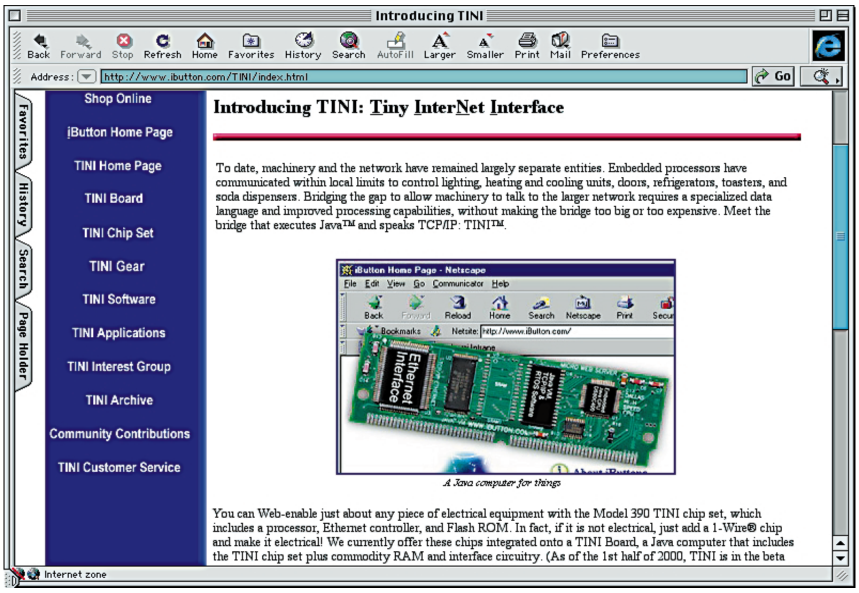
**The chip in the picture above is a complete web-server.
It is about the size of a mere match-head.**

(Photo courtesy of Cris Pedregal-Martin)

The single chip computer in the above picture runs the **iPic web-server**, the world's tiniest implementation of a TCP/IP stack and a HTTP web-server. The chip above is a complete micro-computer, and it includes all components of a complete computer on a single tiny micro-chip (this includes the CPU (central processing unit), memory, serial port interface circuitry, and clock oscillator).

The chip is connected directly to an Internet router, which is essentially the same as an Internet connection from an ISP. When you visit the iPic Web-server, by clicking on this demonstration link, (or this mirror), your web-browser connects to the chip shown in these photographs and the web-pages you see are sent to your web-browser directly from the tiny chip.

*Note: **The serial link to the iPic demo site is currently experiencing a load upto the full link capacity.** The iPic demo link connects to the iPic connected on a serial line. In case the link to the iPic is overloaded, please browse the mirrored files at this mirror .*


Introducing TINI
Back Forward Stop Refresh Home Favorites History Search AutoFill Larger Smaller Print Mail Preferences
Address: http://www.ibutton.com/TINI/index.html
Go
Favorites History Search Page Holder
Shop Online
iButton Home Page
TINI Home Page
TINI Board
TINI Chip Set
TINI Gear
TINI Software
TINI Applications
TINI Interest Group
TINI Archive
Community Contributions
TINI Customer Service
iButton Home Page - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Secu
Bookmarks Netsite: http://www.iButton.com/
Ethernet Interface
Internet zone

Introducing TINI: Tiny InterNet Interface

To date, machinery and the network have remained largely separate entities. Embedded processors have communicated within local limits to control lighting, heating and cooling units, doors, refrigerators, toasters, and soda dispensers. Bridging the gap to allow machinery to talk to the larger network requires a specialized data language and improved processing capabilities, without making the bridge too big or too expensive. Meet the bridge that executes Java™ and speaks TCP/IP: TINI™.

A Java computer for things

You can Web-enable just about any piece of electrical equipment with the Model 390 TINI chip set, which includes a processor, Ethernet controller, and Flash ROM. In fact, if it is not electrical, just add a 1-Wire® chip and make it electrical! We currently offer these chips integrated onto a TINI Board, a Java computer that includes the TINI chip set plus commodity RAM and interface circuitry. (As of the 1st half of 2000, TINI is in the beta plete source code is available.

Microchip, the manufacturer of the PIC family, is also convinced of a great future for its processors.
On *http://www.microchip.com/internet/* you may find much relevant information. Other manufacturers are also active in this field. At *http://world.std.com/~fwhite/ace/* you may find a description of WebAce implemented on a Fairchild ACE1101MT8. Here, too, pictures of the chip and a matchstick.

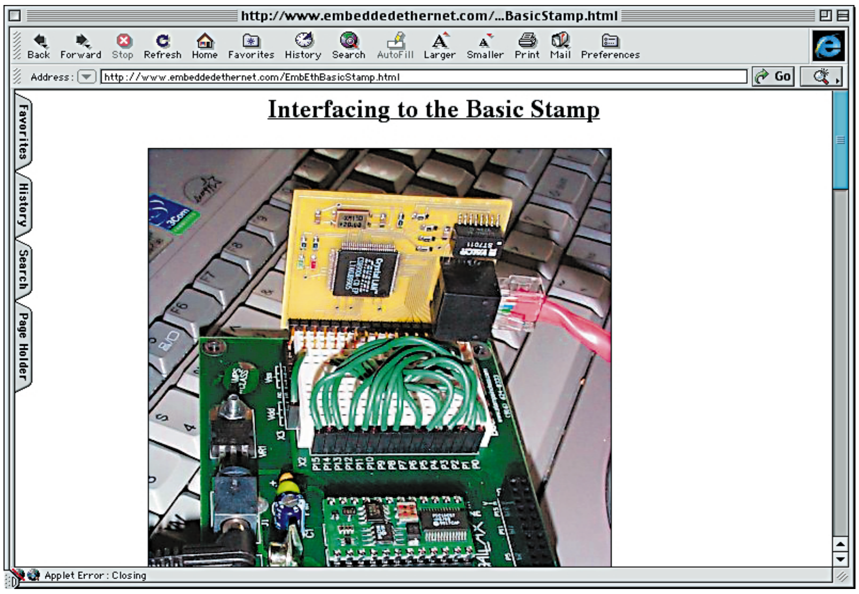
More matchsticks, a box full actually, are found at *http://wearables.stanford.edu/* We saw a complete 486 processor no larger than a matchbox and running a Linux webserver.

In all of the above examples, the physical linking is via RS232. Using SLIP or PPP, TCP/IP is being implemented. As a matter of course, this solution is relatively slow and not very flexible. Because most companies these days employ Ethernet networks, it would be very convenient if embedded equipment would also be fitted with Ethernet connectivity. On *http://www.embeddedethernet.com* a small Ethernet controller is shown, complete with application examples for, among others, the BASIC Stamp.

Currently the nicest interface we feel may be found at
http://www.ibutton.com/TINI/index.html
The TINI (Tiny InterNet Interface) is a printed circuit board the size of a SIMM. It actually holds a processor running a virtual Java machine, together with a real-time operating system ad a TCP/IP stack. The interface with the real world comprises Ethernet, CAN, RS232, I^{2}C, 1-Wire and a number of digital inputs/outputs. TINI is currently in its beta phase and available for $50 if you want to experiment with it. Ultimately TINI will be packed into a single chip product and become available at a much lower price. Naturally, TINI has created a number of websites. At *http://www.systronix.com* a number of multi-purpose TINI interface boards may be found. The unofficial TINI Information Site at *http://www.smartc.com/tini/index.html* provides background information and user applications. Dallas Semiconductor maintains a website for bug reports and other developers' information on *http://tini.dalsemi.com/* Another website,
http://www3.sympatico.ca/guillaume.fournier also contains masses of information in Tini.

In conclusion we'd say that Scott McNealy's vision of the future may be closer than we think and that there are unprecedented possibilities for home constructors.

(005048-1)

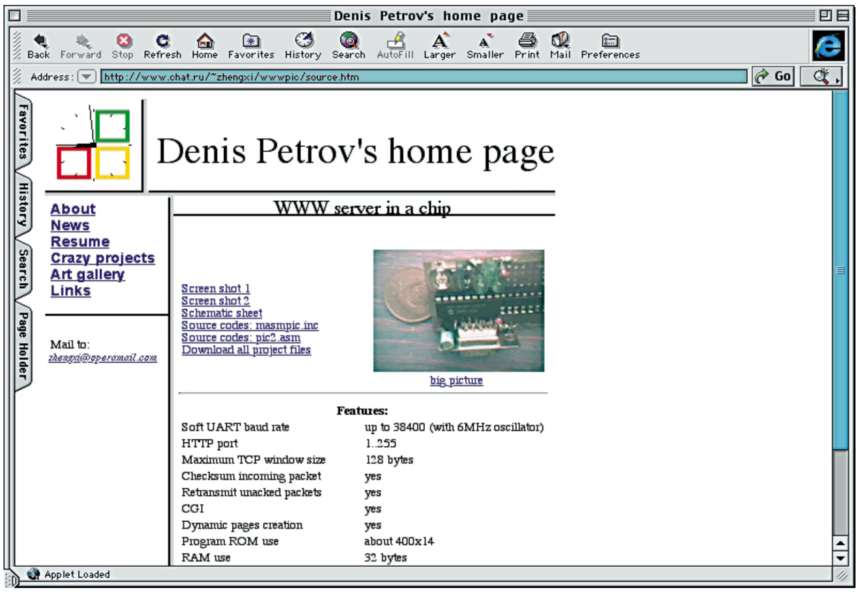


http://www.embeddedethernet.com/...BasicStamp.html

Address: http://www.embeddedethernet.com/EmbEthBasicStamp.html

Interfacing to the Basic Stamp

Applet Error: Closing



Denis Petrov's home page

Address: http://www.chat.ru/~zhengxi/wwwpic/source.htm

Denis Petrov's home page

About
News
Resume
Crazy projects
Art gallery
Links

Mail to:
zhengxi@operamail.com

WWW server in a chip

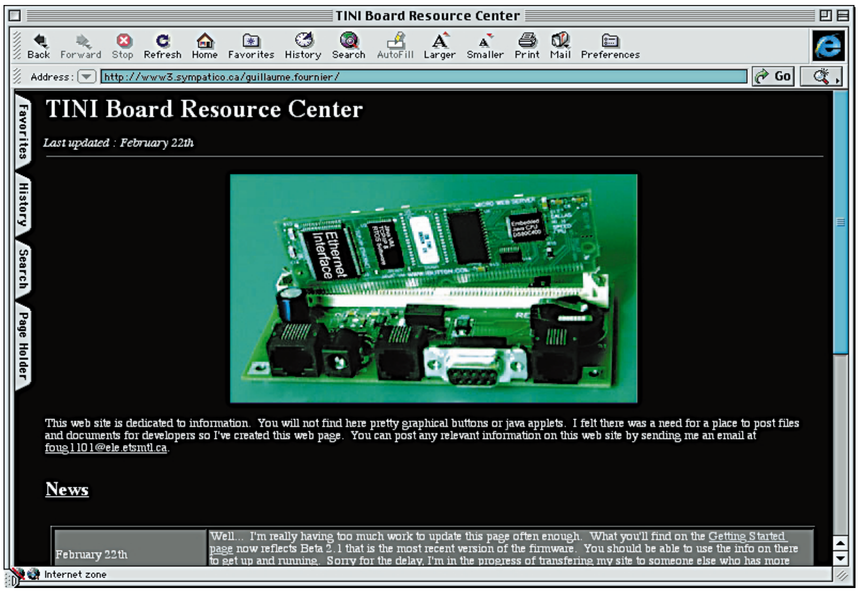
Screen shot 1
Screen shot 2
Schematic sheet
Source codes: masmpic.inc
Source codes: pic2.asm
Download all project files

big picture

Features:

Soft UART baud rate	up to 38400 (with 6MHz oscillator)
HTTP port	1..255
Maximum TCP window size	128 bytes
Checksum incoming packet	yes
Retransmit unacked packets	yes
CGI	yes
Dynamic pages creation	yes
Program ROM use	about 400x14
RAM use	32 bytes

Applet Loaded



TINI Board Resource Center

Address: http://www3.sympatico.ca/guillaume.fournier/

TINI Board Resource Center

Last updated : February 22th

This web site is dedicated to information. You will not find here pretty graphical buttons or java applets. I felt there was a need for a place to post files and documents for developers so I've created this web page. You can post any relevant information on this web site by sending me an email at foug1101@ele.etsmtl.ca.

News

February 22th	Well... I'm really having too much work to update this page often enough. What you'll find on the Getting Started page now reflects Beta 2.1 that is the most recent version of the firmware. You should be able to use the info on there to get up and running. Sorry for the delay, I'm in the progress of transfering my site to someone else who has more

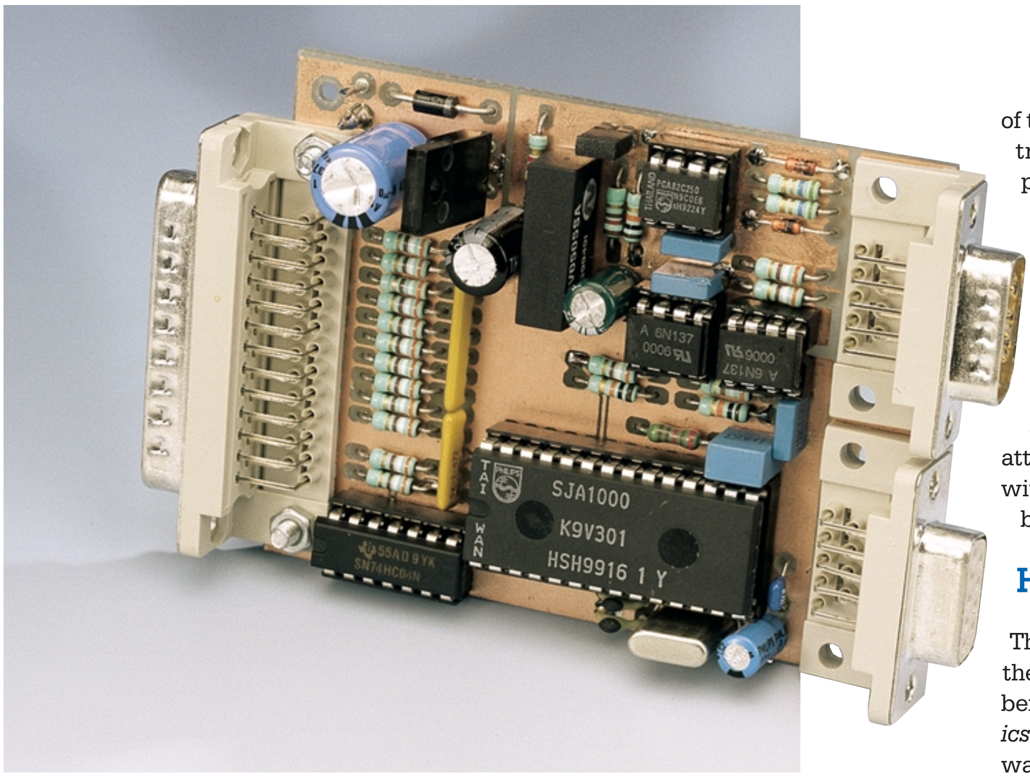
Internet zone

PC Interface for the CAN Bus

with new software

Design by R. Lock

The versatility of a CAN Bus system increases as more equipment of different types are connected to the Bus. Using the interface described here, any PC can be linked to a CAN Bus system.



Before any piece of equipment is designed to connect to a PC, the main question that needs to be answered is: how will the equipment talk to the PC? We could pass the data through a plug-in card, connecting to the PC's motherboard but this would mean that most Notebook PC's would not be able to accommodate it. Using the serial interface is also a possibility but is relatively slow, so the parallel printer interface seems to be the best bet to handle the high data rate of this CAN Bus system. Data will be transferred bidirectionally so the printer interface needs to be configured for this mode. On the printer card of older PC's this is achieved by changing a jumper. On newer Pentium machines, you enter the BIOS setup procedure and select EPP mode. So just think, just about any old PC gathering dust in the attic can be given a new lease of life with the addition of this low-cost, bidirectional, interface card.

Hardware

The circuit diagram is very similar to the project described in the November 1999 edition of *Elektor Electronics* — the main difference is the software which will run on the PC.

The SJA1000 is once again used as the CAN Bus Controller (the PCA82C200 can also be substituted). Its function was fully described in the November 1999 edition of *Elektor Electronics*, and a datasheet can be downloaded from the Philips website at: *www.semiconductors.philips.com*

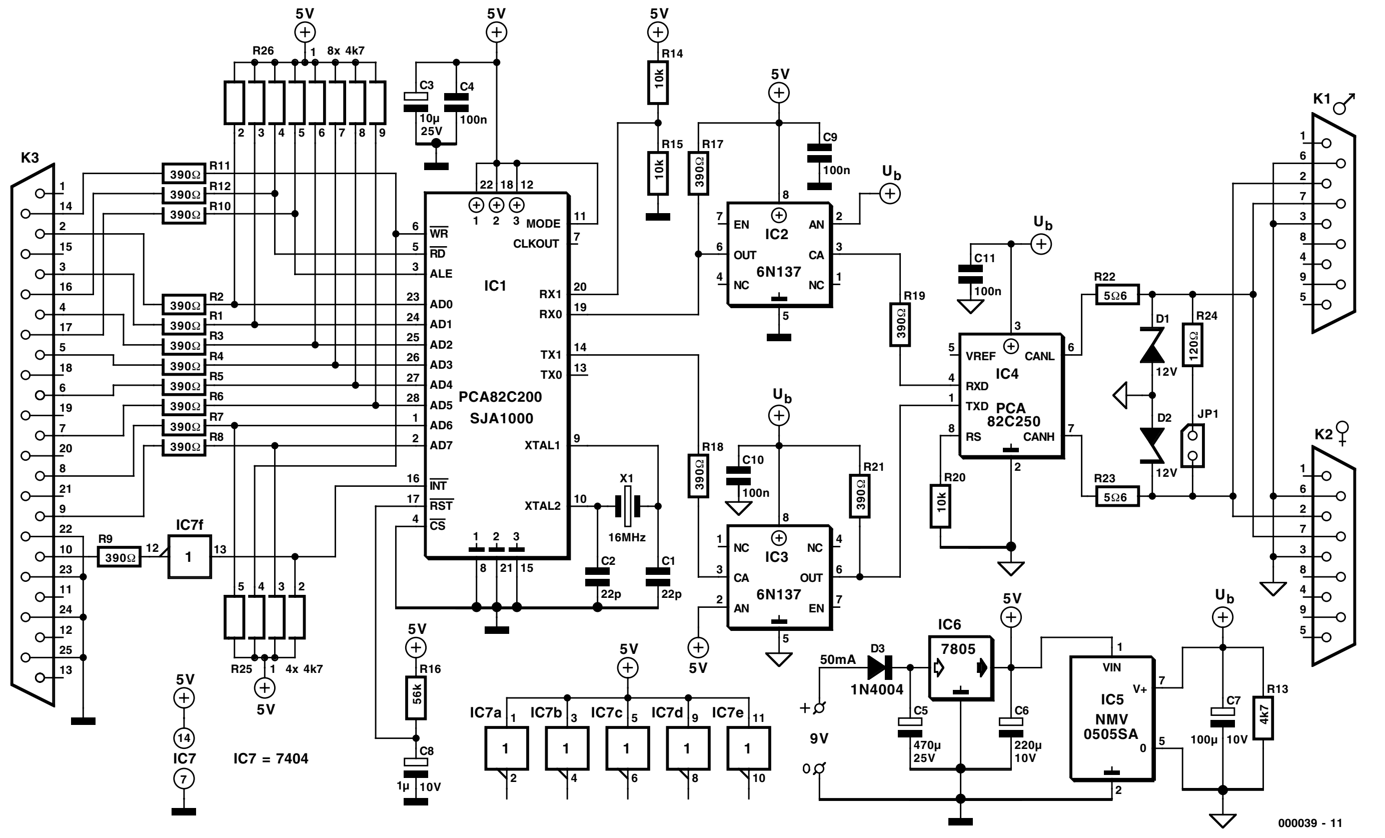


Figure 1. Circuit diagram of the CAN Bus Interface for PC parallel ports.

Although the PCA82C200 and SJA1000 are pin-compatible, voltage divider R14/R15 is only needed for the PCA82C200. If you are using the SJA1000 it is better to connect the RX1 input directly to ground by omitting R14 and replacing R15 with a wire jumper.

Some of the hardware of the controller must also be configured by the software. The internal comparator, for example, needs to be bypassed by setting the CBP (bit 6) of the CDR register. The line drivers are also configured by writing $1A_{HEX}$ to the OCR register.

The SJA1000 is designed to be connected directly to the read and write lines of a microcontroller. In this application it is necessary to generate and read these signals using software on the PC to control the state of the parallel ports signals. The data register of the parallel port (Pins 2-9 of K3) are connected to the multiplexed data/address bus of the SJA1000, while control signals read, write and ALE are generated by the port signals (Autofeed, Init Printer and Select Input). The acknowledge line (pin 10) is used by the SJA1000 as an interrupt source to the PC.

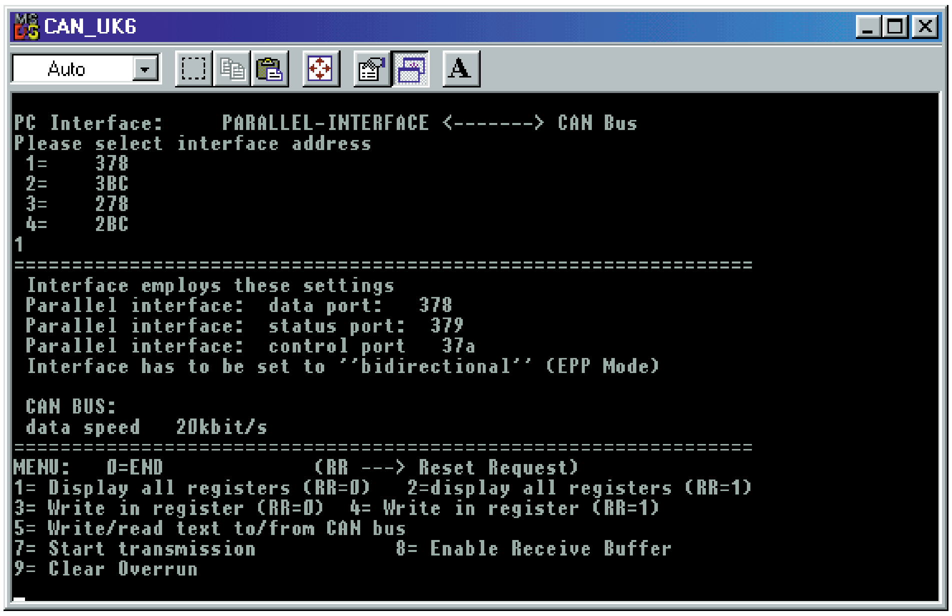
Because the CAN bus is optically coupled to the control circuit via IC2, IC3 and IC5 (a DC-DC converter supplying power for the bus side circuitry), any nasty voltage transients induced in the CAN bus cabling will be prevented from damaging the sensitive interface control circuitry and, eventually, your precious PC. If however you consider the likelihood of such damage to be small then it is possible to omit the DC-DC converter and link the +5 V connections at the input and output sides of the board, and do the same for the two ground connections.

Jumper JP1 allows the 120Ω terminating resistor to be connected across the two wires of the CAN bus. This jumper should only be used for stations at either end of the bus — none of the stations in between should have the jumper fitted. In practice, using this jumper can be a bit awkward if equipment is frequently added or removed from the bus. In those cases, it is better not fit the resistor to the board but instead connect it between pins 7 and 2 of a free sub-D plug, which then acts as a bus termination device. This method ensures that there will never be more than two termination resistors connected to the bus, (important if you don't want to overload the drivers) and they will only fit at the ends!

The wiring of the 9-pin sub-D bus connector is not defined in the CAN standard. As opposed to earlier *Elektor Electronics* projects related to the CAN bus, in this project we not

Features

- Interface for PC parallel port in bidirectional EPP Mode
- CAN Bus connection via 9-way sub-D connectors
- 120 Ω bus termination resistor connected optionally
- CAN-Controller SJA1000 or PCA82C200
- Bus coupling via Transceiver PCA82C250
- Opto-isolators and dc-dc converters for complete electrical isolation
- 9-12V power supply from conventional mains adaptor



```
PC Interface:     PARALLEL-INTERFACE <-------> CAN Bus
Please select interface address
 1=    378
 2=    3BC
 3=    278
 4=    2BC
1
==============================================================
 Interface employs these settings
 Parallel interface:  data port:   378
 Parallel interface:  status port:  379
 Parallel interface:  control port   37a
 Interface has to be set to ''bidirectional'' (EPP Mode)

 CAN BUS:
 data speed   20kbit/s
==============================================================
MENU:   0=END              (RR ---> Reset Request)
1= Display all registers (RR=0)   2=display all registers (RR=1)
3= Write in register (RR=0)  4= Write in register (RR=1)
5= Write/read text to/from CAN bus
7= Start transmission            8= Enable Receive Buffer
9= Clear Overrun
```

Figure 2. Program menu for the CAN controller, condition Reset Request=0.

only swap pins 2 and 7, but also break with the 'rule' that the CAN bus is identical in both directions. In this design you will notice that we have a plug and a mating socket, allowing the interface board to be removed from the bus and simply join the two connectors. Remember that the pin numbering on a socket (i.e., female connector part) is the mirror image of that on a plug (i.e., male connector part).

Software

The DOS program which runs on the PC and drives this interface is available with the source code listings on disk number **006004-1**. Before running this program, it will be necessary to connect the CAN bus interface to the printer port of the PC and connect a supply voltage of 9-12 V to the CAN bus interface. At this point, connection to a CAN bus is not yet necessary.

After starting the program in DOS or through a DOS window you enter the address of the printer port and you will be presented with the program menu. The first menu shows the internal registers of the SJA1000 as three columns (**Figure 2**). The first column (Reg) shows the register number, and the second and third columns show the contents of each register in hexadecimal and ASCII format. If you find that the value shown in all the registers is FF, or the values are 00, 01, 02, 03 etc., then this is a sign that the port has been incorrectly addressed or that it is not working in its bidirectional mode. If this is the case, it is necessary to perform a hardware reset on the system.

With the system working correctly it will now be possible to read all the registers except registers 4-8 because they are only accessible during the initialisation mode when the *Reset Request Bit* (RR) of the Command Register is set to a 1.

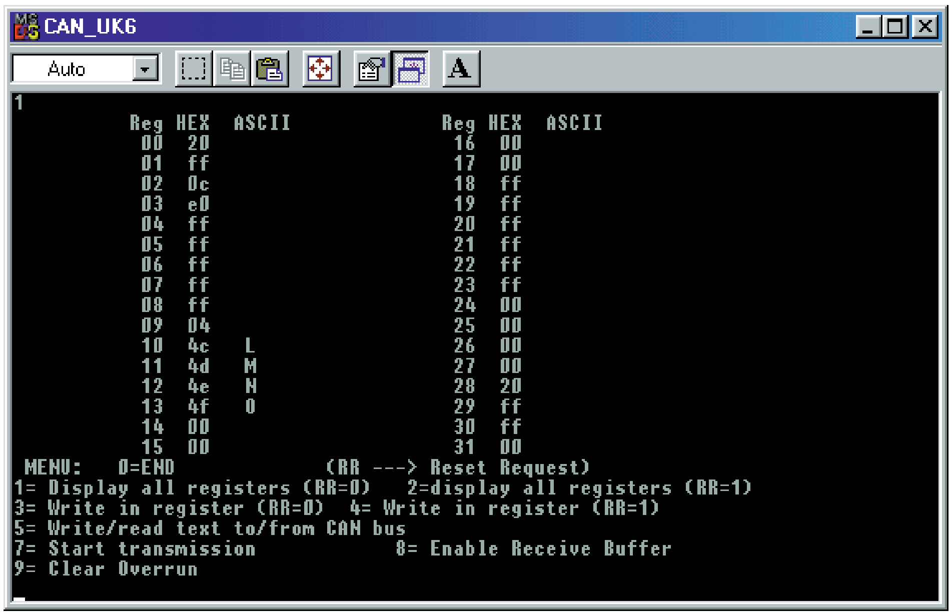
The menu allows the registers to be read and written to by using two different read and write options, one with RR=0 and the other with RR=1. (see box).

Changing the Identifier

An 11-bit message Identifier is contained in register 10 and part of register 11. At initialisation, this Identifier is set to 300 (00100101100_B). If, for example, you want to change this to 512 (01000000000_B) it is necessary to write the 8 most significant bits of the identifier into register 10 (ID3-ID10) and the remaining 3 bits into register 11 (ID0-ID2) at bit position 5, 6, 7. Bit 4 is the *Remote Transmission Request* (RTR) Bit and when this is reset to a zero, a data frame will be sent including the number of data bytes as specified by the data length code. The range of the data byte count is 0 to 8 bytes and is coded as follows:

```
DataByteCount= 8*DLC.3 +
4*DLC.2 + 2*DLC.1 + DLC.0
```

So, if eight bytes need to be sent, then 8_D (00001000_B) must be written into register 11. For reasons of compatibility, no data length code greater than 8 should be used. So for this configuration using menu command 3, 64_D is written into register 10 and 8_D is written into register 11. According to DIN ISO 11898 identifiers in the range 2032-2047 are reserved and should therefore not be used.



```
1
          Reg HEX  ASCII            Reg HEX  ASCII
           00  20                    16  00
           01  ff                    17  00
           02  0c                    18  ff
           03  e0                    19  ff
           04  ff                    20  ff
           05  ff                    21  ff
           06  ff                    22  ff
           07  ff                    23  ff
           08  ff                    24  00
           09  04                    25  00
           10  4c   L                26  00
           11  4d   M                27  00
           12  4e   N                28  20
           13  4f   O                29  ff
           14  00                    30  ff
           15  00                    31  00
 MENU:   0=END              (RR ---> Reset Request)
1= Display all registers (RR=0)   2=display all registers (RR=1)
3= Write in register (RR=0)  4= Write in register (RR=1)
5= Write/read text to/from CAN bus
7= Start transmission            8= Enable Receive Buffer
9= Clear Overrun
```

Figure 3. CAN Controller registers. Registers 4-8 are blocked.

Changing the input filter

When a message is received on the CAN bus interface, an input data filter ensures that only the data frame following a correct CAN bus identifier will be stored. This filtering is controlled by two 8-bit registers: the *Acceptance Code Register* (ACR or Reg 04 in the menu) and the *Acceptance Mask Register* (AMR or Reg 05 in the menu). The 8 most significant bits of the identifier contained in the received CAN message are compared with the values contained in these registers. The AMR tells the controller which bits in the received identifier are relevant. A '0' indicates that it is relevant while a '1' indicates that it is not relevant. For a message to be stored, all **relevant** bits in the received identifier must match their respective bits in

COMPONENTS LIST

Resistors:
R1-R12,R17,R18,R19,R21 = 390Ω
R13 = 4kΩ7
R14,R15,R20 = 10kΩ
R16 = 56kΩ
R22,R23 = 5Ω6
R24 = 120Ω
R25 = 4-way SIL arary 4kΩ7
R26 = 8-way-SIL array 4kΩ7

Capacitors:
C1,C2 = 22pF
C3 = 10µF 25V radial
C4,C9,C10,C11 = 100nF, 5 mm raster
C5 = 470µF 25V radial
C6 = 220µ 10V radial
C7 = 100µ 10V radial
C8 = 1µF 10V or solid MKT raster 5mm

Semiconductors:
D1,D2 = zener diode 12V 400 mW
D3 = 1N4004
IC1 = PCA82C200 or SJA1000 * (Philips)
IC2,IC3 = 6N137 (Toshiba)
IC4 = PCA82C250 (Philips)
IC5 = NMV0505SA (Newport, Farnell #589 810)
IC6 = 7805

Miscelllaneous:
JP1 = Jumper
K1 = 9-way sub-D plug (male), PCB mount.
K2 = 9-way sub-D socket (female), PCB mount.
K3 = 25-waySub-D plug (male), PCB mount
2 solder pins
X1 = 16 MHz quarz crystal
Disk, order code **006004-1** (DOS-Interface and source code in C)
PCB, order code **000039-1**

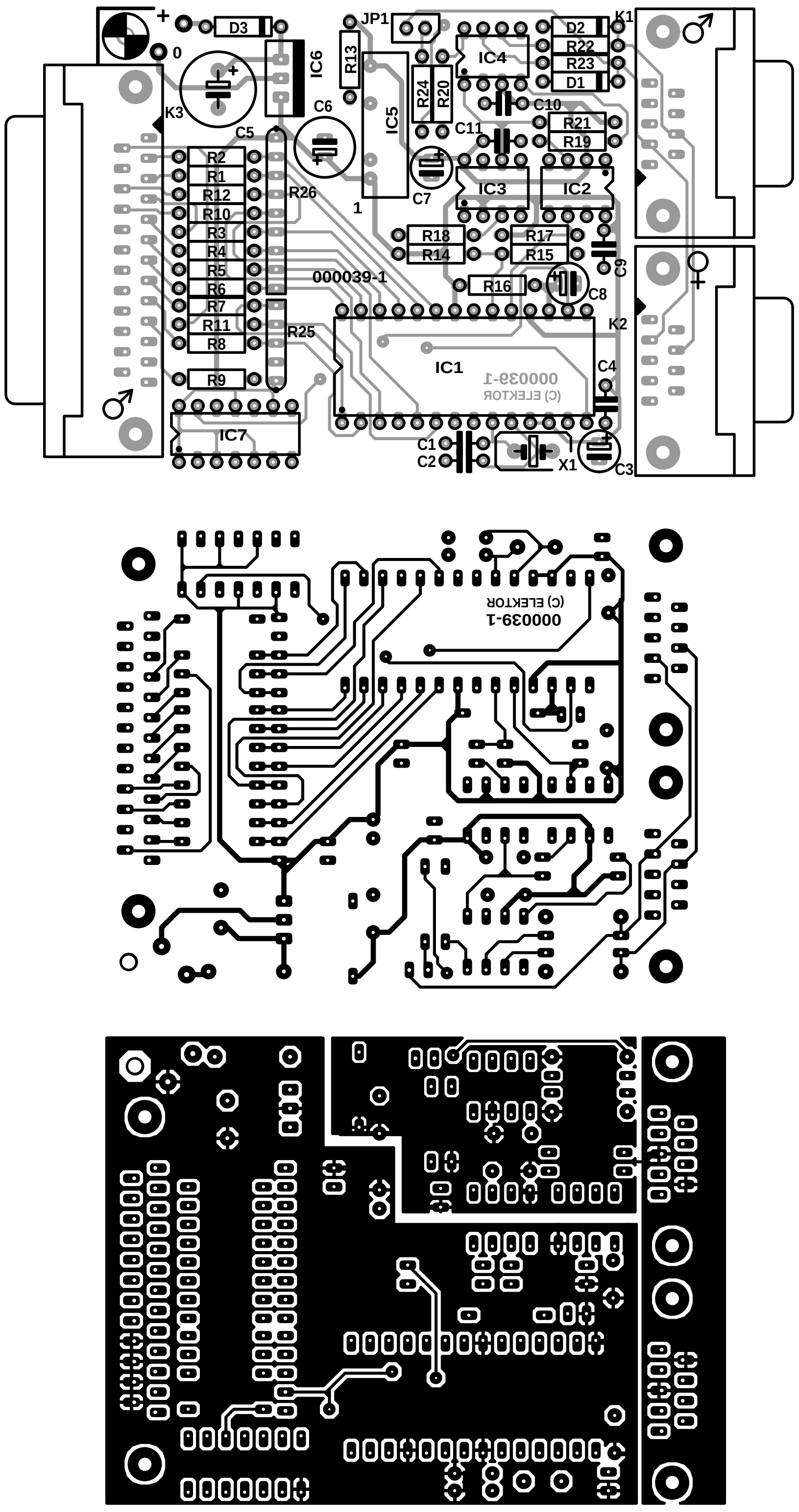


Figure 4. Track layout and component overlay of the PCB designed for the CAN Bus Interface.

Software in C

This driver software allows access to all 32 registers. Using a menu, each register can be written to or read. This allows you to explore the complete communication system. Also included for test purposes are menu options 11, 12 and 13 which give you access to the interface from a PC's register. The PC printer port must be configured for bidirectional operation, either during PC setup (in EPP mode) or by a jumper on the parallel port card. During initialisation, the data rate is fixed at 1Mbit/s.
The driver software has three functions:

1. void init82C200()
This function initialises the CAN Bus Controller for the hardware configuration. The data rate is set to 1 MBit/s.

2. void wr_can(uchar adr, uchar value)
This function allows you to address and write to all of the 32 controller registers.

3. void rd_can(uchar adr)
This function allows you to address and read all of the 32 controller registers.

Example:

init82C200();	called once at the start to initialise the PC interface and the CAN bus controller.
wr_can(12,50);	Writes the value 50 into Register 12 (one of the Transmit buffers).
wr_can(1,10);	Transmit Request. Starts transmission of data in the transmit buffers.
wr_can(1,4);	The receive buffer is released.
wr_can(0,1);	Reset-Request=1. Only some registers can be read and written to in this mode.
wr_can(0,0);	Reset-Request=0. Only some registers can be read and written to in this mode.
unsigned char cw;	
cw=rd_can(22);	Reads Register 22 (One of the receive buffers)
cw=rd_can(2);	Reads the Status Register

the ACR. If for example the AMR is set to FF$_{HEX}$ (all '1'), this will effectively turn off the filtering process by saying that none of the bits of the identifier are relevant, in this case every received message will be stored.

These registers can be read and written to using menu options 2 and 4 respectively after making a reset request (RR=1).

Connecting to a CAN bus

Once the interface has been successfully tested it can be connected to a CAN bus environment. For this you will need at least one other station to connect to, this could be a second PC with this interface fitted.

Menu command number 7 will transmit a message out onto the CAN bus. This message will comprise of the data stored in the transmit buffer together with the identifier and other communication parameters. At the receiving station, the status register (Reg 02) will indicate that a message has arrived by setting bit 0. This can be reset again by command number 8. The controller contains a 2 message deep FIFO receive buffer. If two messages are received but not read by the controller, any further messages will cause a receive buffer overrun to occur and data will be lost. Status register bit SR.1 will be set to indicate this condition. Menu command 9 will reset this bit.

Menu option 5 allows your PC to act as a kind of 'remote display'. In this mode, anything you type on your keyboard will be displayed on any other PC connected to the bus. The source code of this program, written in C, is included in the software, to help you build and develop new applications.

So, all you need to carry out your CAN bus experiments is a PC, two interface cards and two programs to allow you to explore the possibilities of the CAN bus system.

There are three C functions available for communications with the CAN bus controller. These can be added to your C library routines and simply linked to any C program.

The Function
`wr_can(uchar adr, uchar value)`
will write the `value` into the register given by *`adr`*.

The Function
`rd_can(uchar adr)`
returns the value of the register given by *`adr`*.

The following function performs initialisation of the SJA1000:
`initSJA1000()`

(000039-1)

On Project Disk #006004-1:

CAN_UK6	CPP	C source code in English
CAN_PAR6	EXE	DOS Program in German
CAN_PAR6	CPP	C source code in German
CAN_UK6	EXE	DOS Program in English
COPYRI~1	TXT	Copyright notice
CONTENTS	TXT	This text

Literature

[1] W. Lawrentz (ed.)
CAN System Engineering
1997, Springer Verlag, New York

[2] Philips
SJA1000 Stand-alone CAN controller DATA SHEET
www.semiconductors.philips.com

[3] Bosch
CAN Specification version 2.0
Robert Bosch Gmbh, Stuttgart

[4] DIN ISO 11898
DIN, Beuth Verlag, Berlin

Wireless Application Protocol (WAP)

by H. Steeman

The WAP (Wireless Application Protocol) is currently a hot subject in the world of mobile telephony. Almost all operators of mobile telephone networks are developing the infrastructure needed to allow them to offer WAP to their customers.

Information providers, in turn, are busy making existing information suitable for distribution via WAP. The first WAP terminals are already available. Although the Internet has been available to private users for only a few years, it has become an indispensable part of modern society. Previously, you would go to a library to look for background information, but now the Internet is by far the most important source of information. A natural consequence is that more and more users want to have access to the Internet 24 hours a day (at home, at work and while travelling). In order to meet this need, the most obvious solution is to use mobile phones, either on their own or in combination with electronic organisers or personal digital assistants (PDAs). Most organisers, such as the Psion 5mx, the Palm V, the Nokia 9110 and the Compaq Aero, can be equipped with true Web browsers, and some even have Java support. However, in practice it turns out that the limitations of the mobile phone that is used for data transmission, the processing power of the mobile terminal and the resolution of the display do not allow existing Internet pages to be consulted in a usable manner. Consequently, a new standard has been proposed for displaying information from the Internet on mobile equipment: the Wireless Application Protocol.

In practice, the Wireless Application Protocol employs WML pages. These are derived from the HTML pages that are used on the Internet. There are many similarities between the two formats.

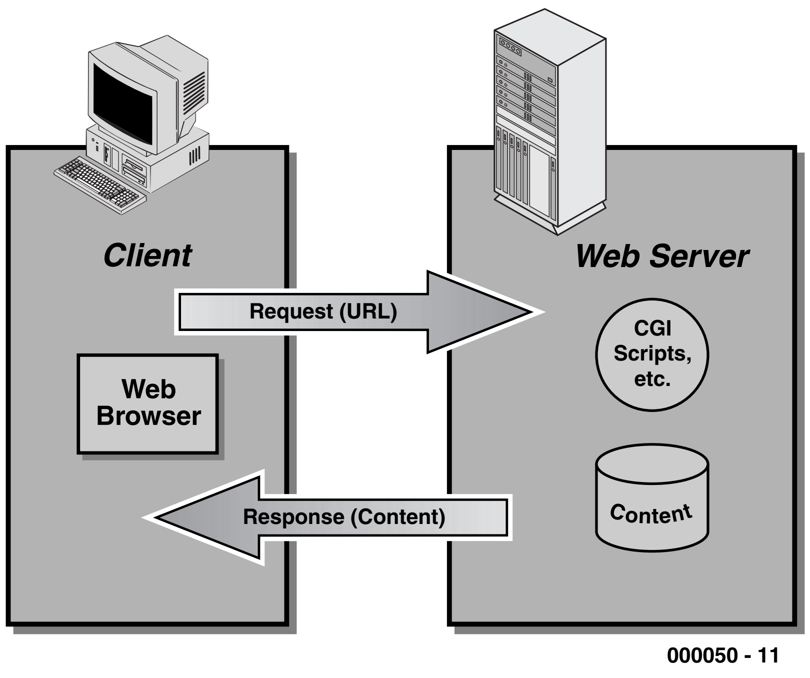


Figure 1. The structure of a WWW session. The client and server communicate with each other using a standardised protocol.

The WWW model

In order to properly understand how the WAP works, you first need to briefly examine how the Internet works (see **Figure 1**). The Internet model is a powerful and extremely flexible system, in which applications that run on a terminal (the client) utilise a standardised series of file formats and communications protocols. Most of these applications have by now been integrated into large, universal terminal programs called web browsers. In practice, a web browser uses a standardised format (such as URL) and protocol (such as HTTP) to send a request for specific information to a log-in server. The log-in server starts a search for the request information and then sends the information to the browser in a standardised format (such as HTML). Several firm agreements are employed to make these data exchanges possible:

- a standard protocol is used for communication between the browser and the server (usually the Hypertext Transport Protocol, or HTTP);
- the sources on the Internet are addressed via Universal Resource Locators (URLs);
- the data formats that are used (for example, for text, sound and images) are clearly defined and should be understood by the browser that is used.

This standardised approach makes it possible to offer users quick and efficient access to information on the Internet.

An Internet user can make contact with the Internet in several different ways. Private users normally choose a dial-up connection to an Internet service provider (ISP). Firms and large offices frequently have fixed data links to the Internet.

In principle, a mobile phone can be used to make a dial-up connection to an Internet service provider, following which the Internet can be consulted. With a GSM connection, the maximum data rate that can be achieved in this manner is 9600 baud. This is too low for practical use, although there are a number of ways in which the situation can be improved, such as data compression and data optimisation. A better solution, which is now becoming available, is the WAP.

The WAP model

The WAP model employs a number of stacked protocol layers to offer the mobile user the desired functionality (see **Figure 2**). The Wireless Application Environment (WAE) is layer is at the top of the communications model. It largely corresponds to the WWW concept. However, in this case the protocol is optimised for use with mobile terminals. It employs the following items:

- Wireless Markup Language (WML), a derivative of HTML that is optimised for use with small terminals that have limited features;
- WMLScript, a simple scripting language comparable to JavaScript, which allows pages to be 'intelligent';
- Wireless Telephony Application (WTA or WTAI), an interface to the telephone functions and the programming environment;
- data formats, including images, text, telephone books and calendar information.

Several lower-level layers are needed to support the use of the WAE model. These are the Wireless Session Layer (WSL), which is the counterpart of the HTTP; the Wireless Transaction Protocol, which is the counterpart of the TCP; Wireless Transport Layer Security (WTLS) and the Wireless Datagram Protocol (WDP), which is the counterpart of the UDP. The combination of all of these layers provides a standardised and secure session. This allows users to perform secure transactions via the WAP, such as making payments. The functionality of WTLS is comparable to that of SSL in an HTTP session. **Figure 3** shows

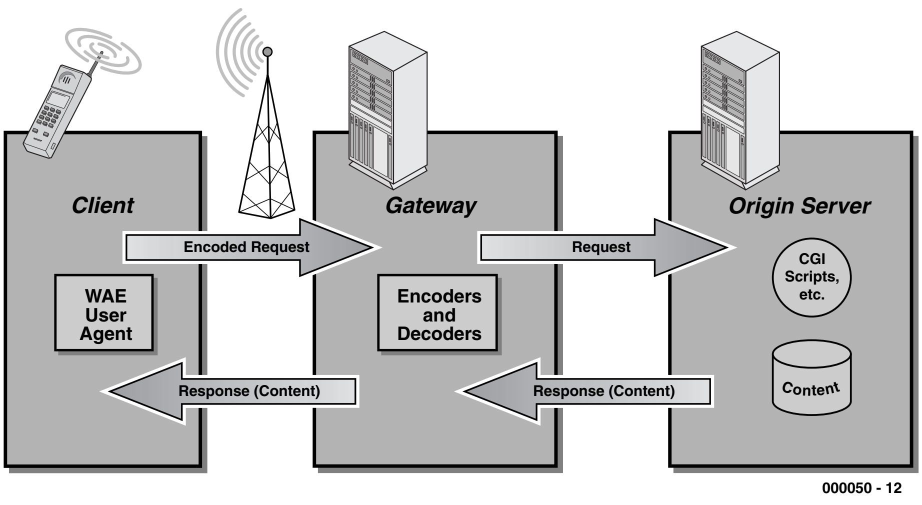


Figure 2. With a WAP connection, the content that the mobile terminal receives is tailored to the characteristics of the terminal. The intermediate gateway translates between the Internet and the world of the mobile network.

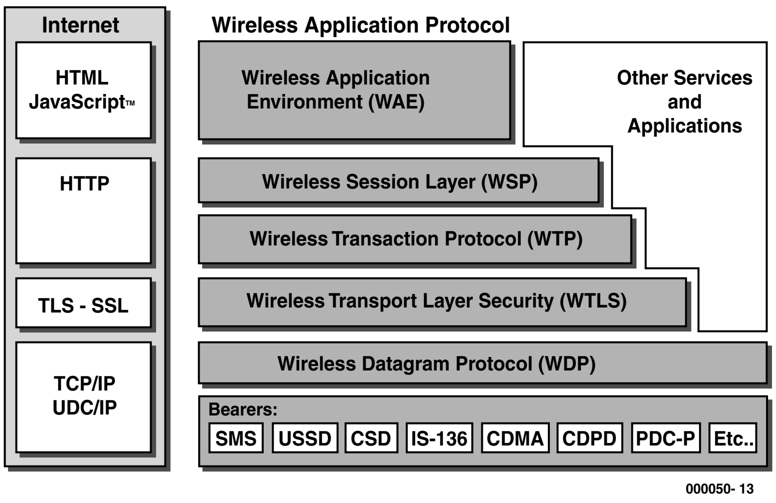

Internet
HTML JavaScript™
HTTP
TLS - SSL
TCP/IP UDC/IP
Wireless Application Protocol
Wireless Application Environment (WAE)
Other Services and Applications
Wireless Session Layer (WSP)
Wireless Transaction Protocol (WTP)
Wireless Transport Layer Security (WTLS)
Wireless Datagram Protocol (WDP)
Bearers:
SMS
USSD
CSD
IS-136
CDMA
CDPD
PDC-P
Etc..
000050- 13

Figure 3. The structure of a 'normal' Internet connection based on the HTTP protocol (left), and a comparable connection using the WAP.

Figure 4. The Nokia 7110, the first telephone on the market that can be used as a stand-alone WAP terminal.

the relationships between the various layers of the WAP and similar layers in WWW traffic. The WAP is presently standardised, and an increasing number of products that comply with the most recent version (WAP 1.2) are appearing on the market.

The WAP ensures that the information that is requested from the Internet perfectly matches the capabilities of the mobile terminal that is used. The graphic information is kept extremely compact, and it is optimised for the browser that is used. If the browser supports colour, the information is delivered in colour. If the browser is limited to black-and-white, the information is adapted to match. By the way, the WAP standard is limited to black-and-white, but some manufacturers extend the standard with support for colour.

The screen resolution supported by the browser, and the limitations of a telephone keypad, are also taken into account as well as possible. This means that a modified telephone (such as the Nokia 7110) can even be used for surfing on the Internet. A thumbwheel (the 'Navi-roller') in this phone the allows the user to surf the net.

Practical considerations

In order to allow the WAP to be exploited as flexibly as possible, use of the protocol can be based on severqal different bearers (services). One of these is part of the GSM standard. Other mobile communication systems can also be used to make WAP sessions possible. These are located in the very bottom layer of the model shown in **Figure 3**.
If we limit our attention to the use of WAP in combination with the GSM network, we see that in addition to a standard dial-up link via a normal speech channel, it is possible to use the Short Message Services (SMS) or a packet-oriented data link (IP traffic). Presently, most operators offer WAP via a dial-up link using the speech channel. This means that a telephone connection is used for the duration of the WAP session, just as with a dial-up connection using the fixed telephone network.

If SMS is selected instead, communications between the client and the server are maintained with the aid of SMS messages. Up to 140 bytes (or 160 characters) can be sent in each message (text is coded using 7 bits in SMS traffic). If this information is used efficiently, the number of SMS messages per session can be kept to a minimum. When SMS is used, it does not matter how much time elapses between two successive actions in the terminal. Only when information is requested or supplied is there any activity on the mobile network (and thus charges). This approach requires the network operator to incorporate an extra server in the network. This server translates the received SMS instructions into tasks that can be processed by the WAP server.

The best option for working with the WAP is to use packet-switched data links. This communications protocol can be used once the mobile telephone network operator has implemented General Packet Radio Services (GPRS). With a GPRS connection, the mobile phone is connected to the network via a packet-oriented data link (IP). This effectively means that the GSM network is a subset of the Internet, and all of its mobile phones are IP terminals. This approach makes very efficient use of the available network bandwidth. For the user, the introduction of GPRS means that his or her telephone is always connected to the mobile network, and thus with the Internet. The fees that are charged in this case are based on the amount of data received, rather than on the amount of time used. GPRS will be implemented in the first mobile networks later this year.

With the combination of WAP and GPRS, it is possible to set triggers using the browser. For example, you could specify that you want to receive a message if the trading price of a particular stock goes above or below a certain level. As soon as the threshold level is reached, a message will appear on the screen of the terminal ('WAP push'). As long as GPRS is not available, the only way to deliver information at a particular time is to use SMS messages. As soon as the WAP is operating, the message is sent via this carrier and appears on the screen.

User information

Before information can be displayed by a WAP browser, it must be put into WML format. Content providers can open special WML sites for this purpose. The mobile surfer is then sent to the correct address by means of a particular URL. A number of providers of translation systems have also appeared on the market. These translators fetch HTML or XML information from the Internet and translate it into WML information. This makes it possible to maintain up-to-date information (such as stock exchange quotes, timetables and travel information) on a single site, while allowing it to be consulted in various manners using different types of platforms. Mobile network operators can incorporate this translation function in their networks, and thus offer 'tailored' information services to their customers.

(000050-1)

DVD – the Megastore

with a great future

By Harry Baggen

After the Digital Versatile Disk (DVD), a marriage of the Super Density CD from Toshiba/Time Warner and the Multimedia CD from Sony/Philips, had been introduced at the Consumer Electronics Show in Las Vegas in early 1996, there was much wrangling among producers, computer companies and film makers to agree on a single world standard. Once that was agreed, DVD players appeared on the market in early 1998, but, originally, their adoption by consumers was slow. This was mainly because there were not many pre-recorded DVD's available. However, that situation has changed and over the past two years the sale of DVD players has really taken off.

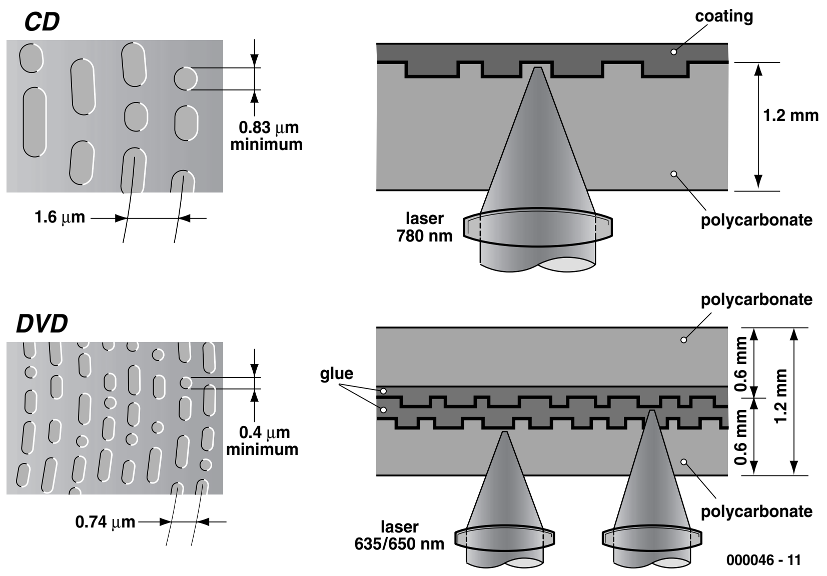


Figure 1. A DVD may consist of up to four layers, two at either side. Shown are the most important physical properties of a DVD compared with those of a CD.

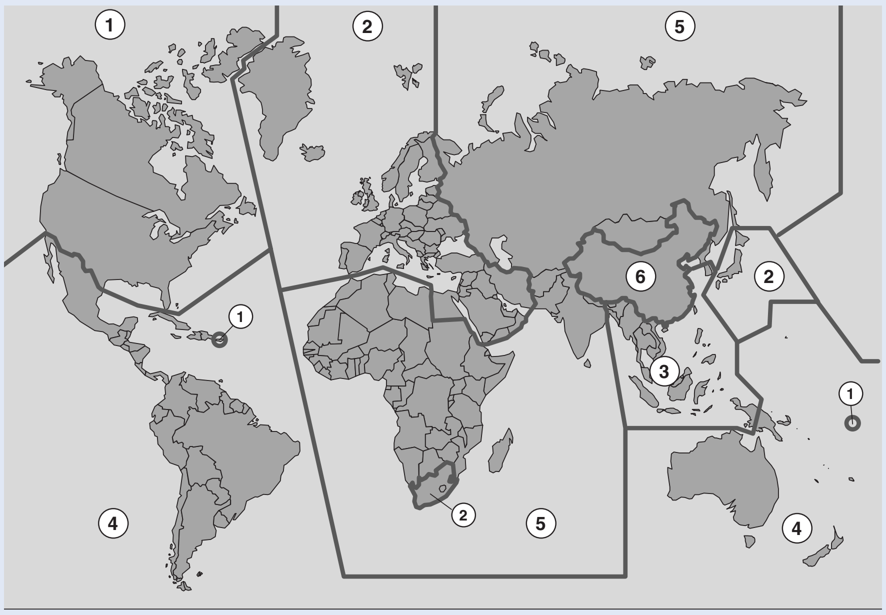
Although the storage capacity of a Compact Disk (CD) at 650 Mbyte appears very large, it is much too small for storing multimedia (combinations of graphics, pictures and sound) data. More especially, it is totally inadequate for storing a complete motion picture with good quality. There have been attempts to store a motion picture onto a CD in MPEG-1 (Moving Pictures Experts Group) format, but this was a complete failure owing to non-availability of suitable domestic players, software, and lack of interest from potential buyers. Moreover, the quality of reproduction was poor and a motion picture of 90–120 minutes duration needed two disks.

The DVD introduced in early 1996 at the Las Vegas Consumer Electronics Show was made possible by improved semiconductor lasers and enhanced CD technologies. With a storage capacity of 4.7–17 Gigabytes, these disks can store up to 11 times as much information as a CD, sufficient for recording a complete, high-quality motion picture, combined with a number of audio channels and some miscellaneous features such as subtitles and simultaneous viewing angles.

In Europe, America and Japan, the popularity and sales of DVD players (for DVD-Video) have risen dramatically over the past year or so. In Britain, sales figures run into many tens of thousands. The main reasons for this acceleration are the growing availability of prerecorded motion pictures, TV Drama, and Concerts, and rapidly falling prices (a good DVD-Video player can be had from about £180 upwards; prices of DVD-Video titles are roughly the same as thos e for similar Video Tapes) . Although Europe is way behind the United States in this respect, at the time of writing (April 2000), there are over 1500 DVD-Video titles available from distributors and producers like HMV, Choices Direct, Towers, and others, in Britain, and this figure is expanding daily. In the

Regional coding

The film industry has had considerable influence on the DVD standard in respect of copy protection and regional protection. The latter gave the industry the opportunity of determining when a certain motion picture would be introduced in a certain region of the world. To this end, the DVD standard contains a regional code for each of six regions of the world. A DVD with one of these codes can be played only on a DVD player with the same code. Already, there are a number of organizations that specialize in converting standard DVD players into players that no longer react to the regional code of a DVD. Computer users are not affected since there are a number of programs that enable changing the regional setting of most software DVD players in a straightforward manner.



000046 - 15

Regional codes

0	whole world
1	USA, Canada
2	Europe, Middle East, South Africa, Japan
3	Southeast Asia, Taiwan
4	Central and South America, Mexico, Australia, New Zealand
5	Russia, Africa (other than South Africa), India, Pakistan
6	China

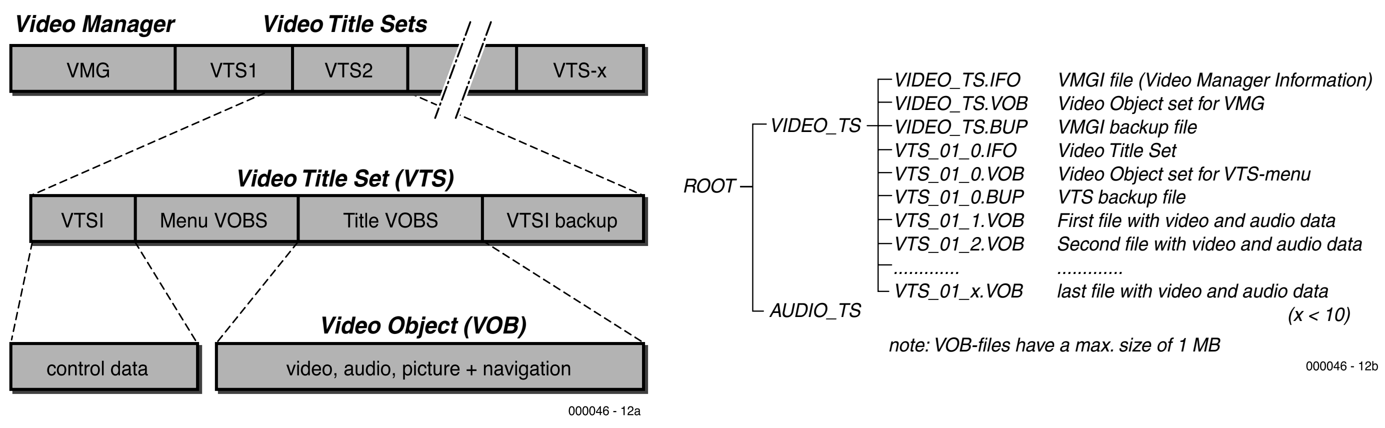

Video Manager
Video Title Sets
VMG
VTS1
VTS2
VTS-x
Video Title Set (VTS)
VTSI
Menu VOBS
Title VOBS
VTSI backup
Video Object (VOB)
control data
video, audio, picture + navigation
000046 - 12a
ROOT
VIDEO_TS
AUDIO_TS
VIDEO_TS.IFO VMGI file (Video Manager Information)
VIDEO_TS.VOB Video Object set for VMG
VIDEO_TS.BUP VMGI backup file
VTS_01_0.IFO Video Title Set
VTS_01_0.VOB Video Object set for VTS-menu
VTS_01_0.BUP VTS backup file
VTS_01_1.VOB First file with video and audio data
VTS_01_2.VOB Second file with video and audio data
....
VTS_01_x.VOB last file with video and audio data (x < 10)
note: VOB-files have a max. size of 1 MB
000046 - 12b

Figure 2. The data structure of a DVD-Video or DVD-Audio is well defined and consists of two directories that contain the various files as shown.

United States, there are over 5000 available, but no doubt the two markets will gradually converge.

Although most new PCs have a DVD slot, the DVD-ROM has been far less successful: most DVD players incorporated in PCs are (mis)used to view films on a monitor via a software decoder. Moreover, there are few computer games available on DVD-ROM and even fewer encyclopedias make use of the medium. On the other hand, some British computer magazines have begun to issue their monthly issues with a DVD-ROM instead of a standard CD-ROM.

The technology

The DVD and CD have much in common: they are the same size (12 cm dia), which makes them identical in appearance, and use the same basic optical storage technology in which information is represented by microscopic pits. However, there the similarities end.

The minimum length of the pits on a CD is 0.83 μm and on a DVD only 0.4 μm. (Because of the smaller size of the pits, the wavelength of the laser used in a DVD player is smaller than that used in a CD player: 635–650 nm instead of 780 nm). This results in a density of 6300 tracks cm^{-1} on a CD and 13,400 tracks cm^{-1} on a DVD. Since the recording surface is 33 mm wide, a CD contains about 20,000 tracks, and a DVD, 44,000 tracks. The distance between adjacent tracks on a CD is 1.6 μm, whereas that on a DVD is 0.74 μm. A CD contains about 7×10^9 bits, whereas a DVD can store up 38×10^9 bits per layer (it has up to four layers which may be recorded separately).

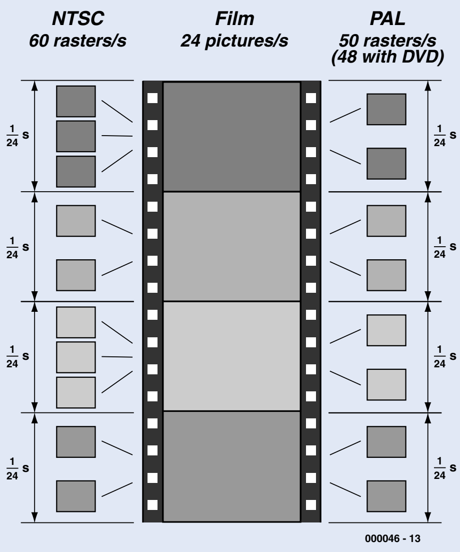
The substrate of a DVD is a 0.55 mm thick disk of polycarbonate, which, because of the shorter wavelength of the laser, is rather thinner than that of a CD. This is covered by a reflective film which contains the pits. Since each reflective layer has a storage capacity of 4.7 Gbyte, constructions of several layers are possible to increase the storage. In the most common construction, two layers are mounted together separated by a distance of 0.5 μm. The lower of these is semi-reflective (30% reflection). When the DVD is played back, the laser starts by reading the first layer from inside to outside, switches over to the second layers, and reads this from outside to inside. This way of operation enables the switchover time of the

From motion picture to video

Converting a motion picture into video throws up the problem that the former consists of 24 frames per second, whereas the NTSC system uses a field frequency of 60 Hz, and the PAL system one of 50 Hz.. This problem is solved as follows.

In the case of NTSC, each motion picture frame is shown alternately during 2 or 3 video rasters. This results in $12 \times 2 + 12 \times 3 = 60$ video rasters or 30 frames per second. Actually, the NTSC field frequency is 59.97 Hz, but the difference is small enough to be negligible.

In the case of PAL, the situation is rather more complex. Converting 24 frames to 50 rasters does not result in a clean ratio. However, an interesting solution for this has been found. Each motion picture frame is shown during 2 video rasters, resulting in 48 rasters or 24 frames per second. When such a motion picture is played back, both film and sound are 4% too fast, resulting in an hour long motion picture lasting only 57.6 minutes. In the case of video, this is deemed acceptable, and the sound is adapted during the production of the DVD (assuming that the manufacturer does his job properly).



laser to be kept to a minimum.

A DVD may also be double-sided with one or two reflective layers on either side. This results in up to four layers with a maximum storage capacity of 17 Gbyte. The manufacturer can therefore produce a single-sided DVD with one or two layers, or a double-sided DVD with 1–4 layers. (See Table)

Type	layers	sides	capacity
DVD-5	1	1	4.7 Gb
DVD-9	2	1	8.5 Gb
DVD-10	1	2	9.4 Gb
DVD-18	2	2	17 Gb

Note that in case of a two-layer construction the full capacity is not available.

Irrespective of the number of layers, a DVD consists of two 0.55 mm thick substrates that are firmly glued together to give an overall thickness of 1.2 mm – the same as a CD.

Currently, three types of DVD are in use:

- DVD-Video for motion pictures and concerts; with a single layer this can store up to 130 minutes of images, three audio surround channels, and four subtitles; with two layers up to 240 minutes of recording may be stored.
- DVD-Audio for high-quality audio recordings. Since the storage format is different from that of a DVD-Video, a modified DVD player is needed.
- DVD-ROM for storing computer data. Apart from pure data, it may also contain a mixture of data, sound and pictures/graphics.

Structure

The layout of a DVD depends on the application. In a DVD-ROM, for instance, the norm and data structure are the same as those of a CD-ROM, that is, to ISO9660. In the case of a DVD-Video or DVD-Audio, a layout is prescribed that must be adhered to strictly by the producer—see **Figure 2**. The Audio_ts file is intended for audio files on a DVD-Audio. Up to now, no producer has made use of this. The audio DVDs on the market are really video DVDs on which the sound has been stored in a video file with a sampling rate of 49 kHz or 96 kHz (often only a still picture is then available).

The VTS (Video Title Set) file contains the control data and video objects of the available menus and titles. A DVD may contain more than one VTS file. The actual pictures, sound, and any miscellaneous matters, are contained in the VOB file. During playback, the DVD machine follows the navigation instructions passed to it by the DVD via the menu structure.

The dvd-video

Since the DVD-Video is by far the most important DVD medium, the reader will no doubt appreciate some more details of this.

The images on a DVD-Video are compressed according to the MPEG-2 standard, otherwise a DVD layer could contain only three minutes of pictures. In MPEG-2, the image resolution is 720×480 pixels in the case of NTSC and 720×576 pixels in PAL. There are also a number of smaller resolutions in the standard, but in the vast majority of DVD-Video the stated resolutions are used.

A single-sided, single-layer disk can contain up to 130 minutes of pictures and sound; with two layers, up to 240 minutes.

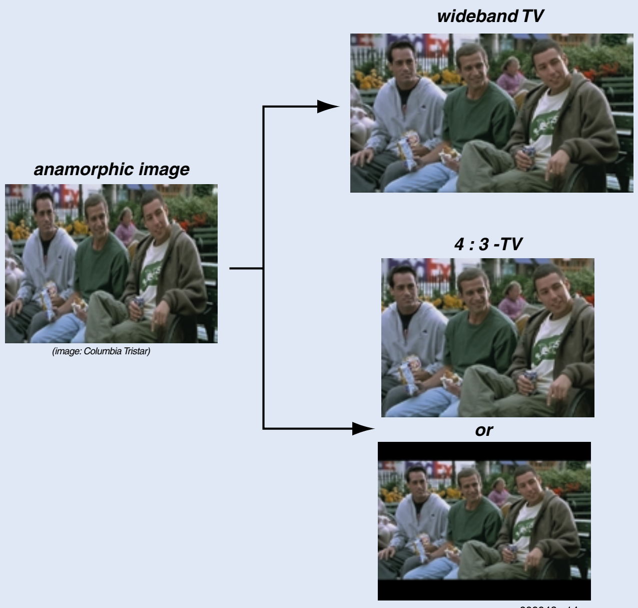
The maximum permissible bit rate is 9.8 Mbit/s, but this is not always used: the average data rate is 4.7 Mbit/s. The actual rate depends, among others, on the number

Bye, bye to the black bars?

A wideband film can be viewed on a standard TV set with 4:3 aspect ratio in a number of ways. The usual way is reproducing the film with the familiar black bars at the top and bottom of the picture (as it were, through a letterbox), but there are alternatives.

Letterbox and pan&scan

Most DVD players have a 'letterbox' setting. When this is enabled, the sides of the picture are cut off so that only the centre portion of the picture is seen. Since this is often inconvenient, not to say annoying, the pan&scan method was developed. In this, the sides of the picture are also cut off, but during the making of the master DVD an indication is given at each frame if the particular frame must be shown to avoid important details getting lost. This method is a good solution to the problem but there are few DVD-Video titles in which the maker has taken the trouble of incorporating it.



Anamorphic or with black bars?

The PAL standard assumes an aspect ratio of 4:3. When a wideband motion picture is viewed, there will be black bars at the top and bottom of the screen. In a widescreen TV set the picture can be enlarged until it fills the screen, but this affects the resolution since a number of picture lines are then not used. Since DVD-Video players have to convert the MPEG-2 data to PAL or NTSC in any case, the manufacturers have devised a trick. Motion pictures are horizontally compressed onto a DVD-Video so that the original wideband image fits in a 4:3 screen. During playback, the viewer can choose how the film is to be shown. In a widescreen TV set the lines are stretched by the DVD player to such an extent that a 16:9 aspect ratio is obtained. In this way, all picture lines are used optimally. In case of a 4:3 TV set, the DVD player converts the data in such a way that a complete picture is shown with black bars at the top and bottom just as is the case with a wideband TV transmission.

Sequential scanning

Computer monitors give a much more stable and clearer picture than a TV screen, since they use sequential scanning, that is, produce the entire picture in a single field (or raster). Most TV systems use a system interlaced scanning. In this, the lines of successive rasters are not superimposed on each other, but interlaced: two rasters form a complete picture or frame. The number of complete pictures per second is the frame frequency, which is half the number of rasters per second, that is, half the field frequency. Since the rasters are accurately interlaced, and the phosphorous film inside the TV tube has a certain delay time, human eye perceives this as a complete picture. The interlacing scanning is necessary owing to the limited bandwidth of the television broadcast channels.
In a computer system, the graphics card sends 60, 75 or even 100, frames per second to the monitor which results in a highly stable picture. Although the material on DVD-Video disks is made up of frames, the DVD player has to convert these into the requisite rasters for the TV set. Fortunately, sequential scanning is beginning to make an entree into the world of television and TV sets processing frames rather than rasters are becoming available, for instance, from Hitachi. Unfortunately, current DVD players must be modified to deliver frames to such a TV set. Manufacturers of current generations of DVD players have realized this too late, but there are specialist firms who can supply suitably modified DVD-Video players.

of audio channels used.

The pictures may be decoded to the NTSC or PAL standard, which affects not only the resolution, but also the number of frames to be reproduced per second. In the case of NTSC, this number is 30, and with PAL, 25. The manner in which pictures at a rate of 24 frames/sec are converted to 30 or 25 video frames is described in a separate box (*From motion picture to video*).

Several aspect ratios may be used, depending on the the kind of film that is to be reproduced. To ensure compatibility between standard 4:3 TV sets and widescreen sets, there are several ways in which the picture can be stored on the DVD, for instance, with 4:3 pictures, with or without black bars (at the top and bottom of a standard set showing wideband film), or by the anamorphic method.

Since there are a number of different formats in use, it is not always possible to faithfully reproduce a picture on a standard or wideband TV receiver (without black top and bottom strokes and without losing a portion of the image).

format	
1.33:1 (4:3)	standard TV
1.66:1	wideband transmission for standard TV sets
1.78:1 (16:9)	wideband TV
1.85:1	wideband cinema format
235:1	super wideband cinema format

The player provides various playback facilities to adapt the picture to the wishes of the viewer and to the aspect ratio of the TV set being used, such as wideband, letterbox, and pan&scan. Even the least expensive DVD-Video player on the market is 16:9/4:3 switchable.

A motion picture may contain up to nine different camera angles; the viewer can, by a push on a knob, view a certain situation on screen from up to nine different angles. The video images at the various camera positions are stored in series in the VOB file. In the case of two viewing angles, a frame of either is available in the file and the viewer can determine which of these he/she wants to be shown.

Subtitles may be considered as a sort of still picture superimposed on the moving pictures. A DVD-Video may contain up to 32 different subtitles (for instance, 32 different languages).

There are also a number of audio features. Most modern motion pictures use surround sound in 5–7 channels and an additional subwoofer channel. Various standards are used: Dolby Digital (also called AC-3), DTS, and MPEG-2 The latter is hardly used nowadays. Dolby Digital and DTS offer roughly the same facilities, but since it uses a different compression method (and a larger data stream), DTS gives a slightly better sound quality according to some experts. However, in Europe, DVD-Video titles with DTS sound are currently virtually unobtainable.

Good stereo sound reproduction may also be obtained with linear two-channel PCM (Pulse Code Modulation) coding. With this, sampling rates of 48 kHz and 96 kHz, and resolutions of 16, 20 and 24 bits are possible.

The film maker may use a number of camera angles, various sound channels, several subtitles, and some other facilities, as long as the data rates just stated are not exceeded.

Each DVD-Video is copy-protected via a Content Scrambling System (CSS) which senses whether the relevant playback machine is permitted to read the disk. However, this supposedly watertight system was cracked last year by a 15-year old schoolboy. This is, however, no tragedy because the requisite hardware and computing power for copying a DVD-Video is appreciable, apart from the fact that there are currently no affordable recording machines with the same capacity as the DVD available.

A number of DVDs are also fitted with Macrovision protection that causes interference when a DVD is being copied to an analogue video cassette recorder (VCR).

The future

In the foreseeable future, the DVD-Video will no doubt become the mainstay of prerecorded feature films, TV Drama and Comedy, Musicals, and Documentaries; already, in Britain there are more than 1500 titles available and more are added daily.

What is currently lacking to make the takeover from the VCR complete is a suitable recorder. These have recently gone into production with various manufacturers and some are already available in the shops: the DVD-RW from Pioneer; the DVD-RW from Philips; and the DVD-RAM from Panasonic. It will, as usual, take some time before the manufacturers have agreed on a standard.

Currently, prices are high: about £1500–2000 ($US2000–2700) for a recorder and £15–20 ($US20–27) for a re-recordable DVD. However, these prices are likely to tumble before very long when production (after standardization) gets into full swing. Once that is the case, the days of the ageing VCR are numbered.

(000046-I)

LEGO RCX

ELEKTOR ELECTRONICS

Parameter Table

DATASHEET 6/2000

Parameter Table #1/2 (RCX only)											
Command	**Source** (number in cells below)										
	Var. (0)	Timer (1)	Const. (2)	Motor Status (3)	Random No. (4)	Tacho Counter C(5)	Tacho Speed C(6)	Motor Current C(7)	Prgm-No. R(8)	Sensor Value (9)	Sensor Type (10)
On (MotorList) **Off** (MotorList) **Float** (MotorList) **SetFwd** (MotorList) **SetRwd** (MotorList) **AlterDir** (MotorList) **SetPower** (MotorList, Source, Number)		•		•	•	•	•	•	•	•	•
SetEvent (Source, Number, Time) **ClearEvent** (Source, Number , Time)	0	•	•	•	•	•	•	•	•	•	•
Poll (Source, Number)	0-31	0-3	•	0, 1, 2	•	0, 1	0, 1	2	x	0, 1, 2	0, 1, 2
SerVar (VarNo, Source, Number)	0-31	0-3	–32768-32767	0, 1, 2	1-32767	0, 1	0, 1	2	•	0, 1, 2	0, 1, 2
SumVar (VarNo, Source, Number) **SubVar** (VarNo, Source, Number) **DivVar** (VarNo, Source, Number) **MulVar** (VarNo, Source, Number) **SgnVar** (VarNo, Source, Number) **AbsVar** (VarNo, Source, Number) **AndVar** (VarNo, Source, Number) **OrVar** (VarNo, Source, Number)	0-31	•	–32768-32767	•	•	•	•	•	•	•	•
Loop (Source, Number)	0-31	•	0-255	•	1-255	•	•	•	•	•	•
While (Source1, Number1, RelOp, Source2, Number2) **If** (Source1, Number1, RelOp, Source2, Number2)	0-31	0 - 3	–32768-32767	0, 1, 2	•	0, 1	0, 1	2	•	0, 1, 2	0, 1, 2
Wait (Source, Number)	0-31	•	1-32767	•	1-32767	•	•	•	•	•	•
DatalogNext (Source, Number)	0-31	0-3	•	•	•	•	•	•	•	0, 1, 2	•
SelectDisplay (Source, Number)	0-31	•	0-255	•	•	•	•	•	•	•	•
SendPBMessage (Source, Number)	0-31	•	0-255	•	•	•	•	•	•	•	•

LEGO OCX

ELEKTOR ELECTRONICS

OCX Command Set

DATASHEET 6/2000

Source:
Lego, http://www.legomindstorms.com

Example Application
LEGO Robotics Invention System,
Elektor Electronics April-June 2000.

Communication control commands:

☞	O	Bool	InitComm()
☞	O	Bool	CloseComm()
☞	O	Variant	GetShortTermRetransStatis-tics( )
☞	O	Variant	GetLongTermRetransmitSta-tistics( )
☞	O	Bool	SetRetransmitRetries(immidiateRetries, download-Retries)
☞	O	Bool	IgnDLerrUntilGoodAn-swer()

Firmware control commands:

☞		BSTR	UnlockPBrick( )
☞		BSTR	UnlockFirmware(Unlock-String)
☞		Bool	DownloadFirmware(File-Name)

Diagnostics commands:

☞		Bool	PBAliveOrNot()
☞	O	Bool	TowerAndCableConnect-ed()
☞	O	Bool	TowerAlive()

PBrick system commands:

☞	✎	Bool	SelectDisplay(Source, Number)
☞	✎	Bool	SetWatch(Hours, Min)
☞		Bool	PBPowerDownTime(Time)
☞	✎	Bool	PBTurnOff()
☞	✎	Bool	PBTxPower(Number)
☞	✎	Bool	PlayTone(Frequency, Time)
☞	✎	Bool	PlaySystemSound(Number)
☞	✎	Bool	ClearTimer(Number)
✎		Bool	SendPBMessage(Source, Number)
✎		Bool	ClearPBMessage( )

PBrick output control commands:

☞	✎	Bool	On(MotorList)
☞	✎	Bool	Off(MotorList)
☞	✎	Bool	Float(MotorList)
☞	✎	Bool	SetFwd(MotorList)
☞	✎	Bool	SetRwd(MotorList)
☞	✎	Bool	AlterDir(MotorList)
☞	✎	Bool	SetPower(MotorList, Source, Number)
✎		Bool	Wait(Source, Number)

PBrick input control commands:

☞	✎	Bool	SetSensorType(Number, Type)
☞	✎	Bool	SetSensorMode(Number, Mode, Slope)
☞	✎	Bool	ClearSensorValue(Num-ber)

PBrick program control commands:

☞		Bool	SelectPrgm(Number)
☞		Bool	DeleteTask(Number)
☞		Bool	DeleteAllTasks( )
☞		Bool	DeleteSub(Number)
☞		Bool	DeleteAllSubs( )

PBrick program execution commands:

☞	✎	Bool	StartTask(Number)
☞	✎	Bool	StopTask(Number)

LEGO OCX

ELEKTOR ELECTRONICS

OCX Command Set

DATASHEET 6/2000

☞	✎		Bool	StopAllTasks( )
	✎		Bool	GoSub(Number)

PBrick flow control commands:

✎			Bool	Loop(Source, Number)
✎			Bool	EndLoop()
✎			Bool	While(Src1, No1, RelOp, Src2, No2)
✎			Bool	EndWhile()
✎			Bool	If(Src1, No1, RelOp, Src2, No2)
✎			Bool	Else()
✎			Bool	EndIf()
☞		O	Bool	BeginOfTask(Number)
☞		O	Short	EndOfTask( )
☞		O	Short	EndOfTaskNoDownload( )
☞		O	Bool	BeginOfSub(Number)
☞		O	Short	EndOfSub( )
☞		O	Short	EndOfSubNoDownload( )

PBrick arithmetic/logical commands:

☞	✎		Bool	SetVar(VarNo, Source, Number)
☞	✎		Bool	SumVar(VarNo, Source, Number)
☞	✎		Bool	SubVar(VarNr, Source, Number)
☞	✎		Bool	DivVar(VarNr, Source, Number)
☞	✎		Bool	MulVar(VarNr, Source, Number)
☞	✎		Bool	SgnVar(VarNr, Source, Number)
☞	✎		Bool	AbsVar(VarNr, Source, Number)
☞	✎		Bool	AndVar(VarNr, Source, Number)
☞	✎		Bool	OrVar(VarNr, Source, Number)

PBrick query commands:

☞		O	Bool	SetEvent(Source, Number, Time)
☞		O	Bool	ClearEvent(Source, Number)
☞			Short	Poll(Source, Number)
☞			Short	PBBattery()
☞			Variant	MemMap( )

PBrick data acquisition commands (RCX only):

☞			Bool	SetDatalog(Size)
☞	✎		Bool	DatalogNext(Source, Number)
☞			Variant	UploadDatalog(From, Size)

ActiveX control commands:

☞		O	Bool	SetThreadPriority(threadNo, threadClass,ThreadPriority)
☞		O	Void	GetThreadPriority(threadNo, threadClass, ThreadPriority)

ActiveX event dispatch interface:

A	VariableChange(Number, Value)
A	DownloadDone(ErrorCode, TaskNo)
A	DownloadStatus(timeInMS,sizeInBytes, taskNo)
A	AsyncronBrickError(Number,Description)

Nomenclature:

☞ = Immediate Command.
✎ = Downloadable Command.
O = ActiveX (OCX) command, nothing transmitted to the PBrick
A = ActiveX asynchronous events

LEGO RCX

ELEKTOR ELECTRONICS

Parameter Table

DATASHEET 6/2000

Parameter Table #2/2 (RCX only)

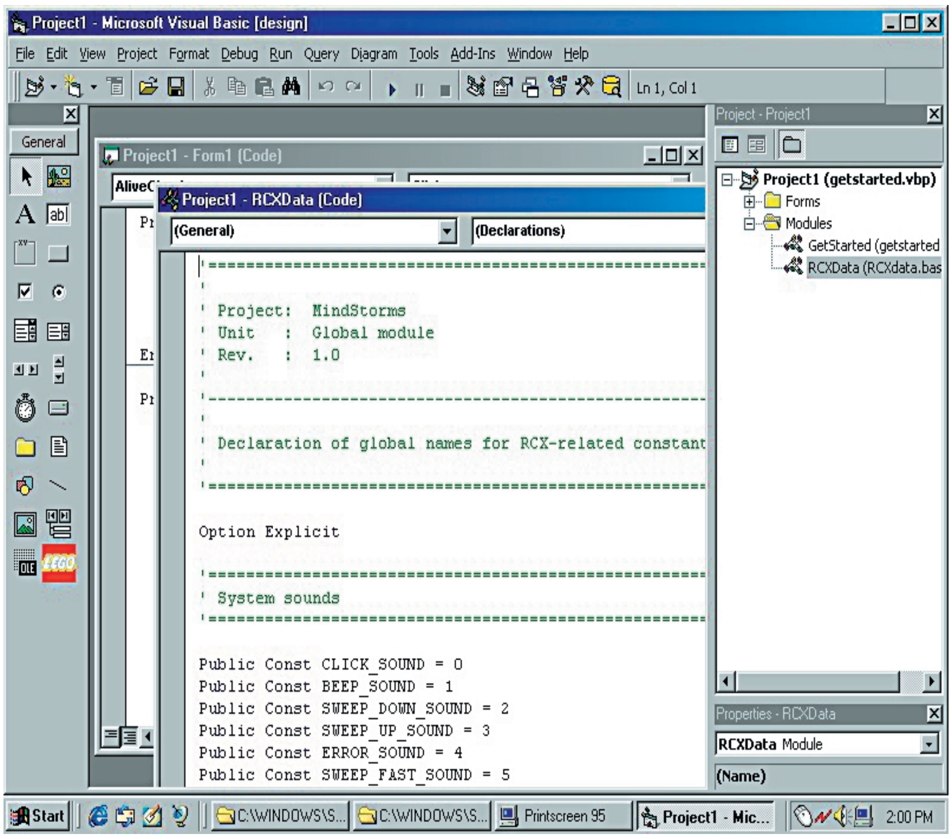
Command	**Source** (number in cells below)						**Motor List**	**VarNo**	**RelOp** > 0 < 1 = 2 <> 3	**Time**
	Sensor Mode (11)	Sensor Raw R(12)	Sensor Bool. R(13)	Watch (R14)	PB Message R(15)	AGC C(16)				
On (MotorList) **Off** (MotorList) **Float** (MotorList) **SetFwd** (MotorList) **SetRwd** (MotorList) **AlterDir** (MotorList) **SetPower** (MotorList, Source, Number)	•	•	•	•	•	•	0, 1, 2	•	•	•
SetEvent (Source, Number, Time) **ClearEvent** (Source, Number , Time)	•	•	•	•	•	•	•	•	•	0-10,000 ms
Poll (Source, Number)	0, 1, 2	0, 1, 2	0, 1, 2	0	0	0	•	•	•	•
SerVar (VarNo, Source, Number)	0, 1, 2	0, 1, 2	0, 1, 2	0	0	•	•	0-31	•	•
SumVar (VarNo, Source, Number) **SubVar** (VarNo, Source, Number) **DivVar** (VarNo, Source, Number) **MulVar** (VarNo, Source, Number) **SgnVar** (VarNo, Source, Number) **AbsVar** (VarNo, Source, Number) **AndVar** (VarNo, Source, Number) **OrVar** (VarNo, Source, Number)	•	•	•	•	•	•	•	0-31	•	•
Loop (Source, Number)	•	•	•	•	•	•	•	•	•	•
While (Source1, Number1, RelOp, Source2, Number2) **If** (Source1, Number1, RelOp, Source2, Number2)	0, 1, 2	0, 1, 2	0, 1, 2	0	0	•	0	•	0-3	•
Wait (Source, Number)	•	•	•	•	•	•	•	•	•	•
DatalogNext (Source, Number)	•	•	•	0	•	•	•	•	•	•
SelectDisplay (Source, Number)	•	•	•	•	•	•	•	•	•	•
SendPBMessage (Source, Number)	•	•	•	•	•	•	•	•	•	•

Lego Robotics Invention System

part 3: working with Visual BASIC

By Luc Lemmens – Email: techdept@segment.nl

In the first two articles in this series, we have become acquainted with the hardware and software of the Lego Robotics Invention System. Here, in the third article, we see how the RCX block can be programmed using Microsoft Visual BASIC or another object-oriented programming language. Such a language puts us a lot closer to the hardware, so that we can better determine what the robot does.



```
'==========================================================
'
' Project:  MindStorms
' Unit    :  Global module
' Rev.    :  1.0
'
'----------------------------------------------------------
'
' Declaration of global names for RCX-related constant
'
'==========================================================

Option Explicit

'==========================================================
' System sounds
'==========================================================

Public Const CLICK_SOUND = 0
Public Const BEEP_SOUND = 1
Public Const SWEEP_DOWN_SOUND = 2
Public Const SWEEP_UP_SOUND = 3
Public Const ERROR_SOUND = 4
Public Const SWEEP_FAST_SOUND = 5
```

In the previous articles, we have seen that the Robotics Invention System (RIS) is a very compact package, in regard to both the hardware and the software. The software is very user-friendly, and it should be perfect for the beginner in robotics and programming. However, if you really want to see what is happening inside the RCX block, you will soon feel the need for other software packages. In this article, we assume that you have already worked with Visual BASIC. If not, you can find all sorts of tutorials on the Internet. A very usable course can be found at *http://emhain.wit.ie/~p98ac25*. This provides step-by-step directions for working with Visual BASIC (VB), and also describes how VB can be applied to the Lego RCX. Most of the tips and tricks described at this site can be directly carried over to Delphi and Visual C++.

When the RIS software is installed, the ActiveX module Spirit.OCX is also installed on the hard disk. The Lego programming software utilises this module, but it can also be used by 32-bit versions of high-level programming languages, such as Visual BASIC, Delphi and Visual C++. This means that we can use these development tools to have a go at the RCX. The text box explains how to link this OCX to the latest versions of these languages.

Linking Spirit.OCX

To link Spirit.OCX to Visual BASIC version 6:

– In the Project menu, select ‘Components’.
– On the Control tab card, check (enable) the option ‘LEGO Pbrickcontrol OLE control module’.
– The Toolbox will now contain a Lego logo that can be placed in a form.

To link Spirit.OCX to Visual C++ version 6:

– In the File menu, select ‘New’.
– Enter a name for the project, and select the application type ‘MFC AppWizard (EXE)’.
– In the following window, select ‘Dialogue based’.
– Click on ‘Finish’.
– In the Project menu, select ‘Add To Project / Components and Controls’.
– Double click on the folder ‘Registered ActiveX Controls’. In the folder, select the short-cut to Spirit Control, click on ‘OK and close the Components and Controls Gallery.

To link Spirit.OCX to Inprise Delphi version 5:

– In the Component menu, select ‘Import ActiveX Control’.
– In the list, select ‘LEGO Pbrick control, OLE control module (Version 1.0)’.
– Click on ‘Install’, and then on ‘OK’.
– Spirit.OCX will now be shown on the ActiveX control tab card.

The RCX as an object

Visual BASIC is an object-oriented programming language, so it shouldn’t be surprising that it sees the RCX as an object with its own properties and methods. These are summarised in the datasheet at the back of this issue. If you’re the type of inquisitive reader that wants to know all the ins and outs of everything, we suggest that you visit the Lego MindStorms Internet site. At *http://www.lego-mindstorms.com/* you will find a link to a software development kit (SDK), which is a somewhat misleading name for two zip files: Pbrick.zip and GetStart.zip. The first file contains Pbrick.pdf, which is a PDF file that contains more than 100 pages of extensive descriptions of the functions of the Spirit.OCX Active-X module. The second file contains a simple demonstration application, written in Visual BASIC, that illustrates nearly all of the programming tricks. Later on, we will have a good look at this.

GetStart.zip also contains the file RCXData.bas, in which all constants for the RCX are assigned logical names. Naturally, these names can only be used in a project if this module is linked in.

There are three ways you can program the RCX using an object-oriented programming language. The first of these, which probably will quickly become boring, is to directly control the RCX from a PC. This allows you to switch the motors on or off, for example, or to read out the sensors and call up the value of the battery voltage. This is all very nice, but if the robot goes out of range of the tower, it will simply carry on as it was, without any control. In this mode of operation, the RCX acts only as a glorified interface between the PC and the sensors and motors of the robot, while the real intelligence sits in the PC.

The second method, which offers many possibilities that we already find in the Lego software, is to develop a program on the PC and then download it to the RCX via the tower. This of course requires more knowledge of the Lego processor system, and we will discuss it shortly.

The third method, which is a mix of the first two methods, involves interactions between a program in the PC and the software in the RCX. This is of course the cat’s whiskers, but it naturally has the same limitation as the first method, which is that the robot must either stay within the transmission and reception range of the PC, or it must be intelligent enough to find its own way back to the PC. However, before you even start to *think* about the attractive possibilities of interactive operation, you must master the first two methods. Consequently, in this article we only look at programming methods.

Method 1: direct control from the PC

If you load the project GetStart.vbp into Visual BASIC, you will immediately see Spirit.OCX appear in the Toolbox with a Lego logo. If you look at the form for this application, you will see that it is a very simple sample application. It is worthwhile to print out the source code and study it carefully, since it will right away give you a good idea of how to program and control the RCX using this language.

For the direct control method, the uppermost six buttons and the four labels on the form are the most important. The ‘Download Program’ button applies to the second method.

The buttons that we are interested in here provide basic communications with the RCX. The topmost button checks whether it is possible to communicate with the RCX (in other words, whether the tower is connected to the PC and the RCX is ‘awake’), the second button requests the software version number, and so on. None of this is especially remarkable, and similar functions are present in the Lego software. For example, you could add buttons to allow the motors to be switched on or the sensors to be read out.

Method 2: downloading a program from the PC

This is what we’re actually interested in: developing a program in Visual BASIC and downloading it to the program memory of the RCX.

As you know, the RCX can hold five different programs, which can be selected using the grey ‘Prgm’ button on the module. But what is not apparent in the Lego software is that each program block can be subdivided

Table 1.
Identification numbers for measurement values in a data logging session.

Type number		Sequence number
0	VAR	0 - 31
1	TIMER	0 - 3
9	SENVAL	0, 1, 2
14	WATCH	0

Table 2.
Sensor types and identification numbers.

Number	Constant	Sensor type
0	NO_TYPE	none
1	SWITCH_TYPE	switch
2	TEMP_TYPE	temperature
3	LIGHT_TYPE	light
4	ANGLE_TYPE	angular

into eight subroutines and ten tasks. The subroutines can be called from all tasks. Each task carries out one specific job, such as reading in a sensor or driving a motor. Whenever a program is started, task 0 is automatically activated, and it usually looks after starting any other tasks that may be present. We thus have a multitasking system, with up to ten tasks that can be executed in parallel. Each program block must contain at least one task, and subroutines are optional.

If you look at the source code for the 'Download Program' button, you will recognise this structure in a very simple example. The first thing that happens is that a program block is selected using the 'SelectPrgm' method, and the value '0' is assigned to the constant MotorControl at the beginning of the program. Note that program blocks are numbered 1 through 5 in the RCX, while in Visual BASIC they are numbered 0 through 4. Next, task 0 is defined, with the logical name 'MotorOnOff'. The RCX functions (such as PbrickCtr.Wait) are always called within this task.

As already noted, this is just a simple example with only one task. A more complicated program will have more than one task, and task 0 essentially only starts the other tasks, in addition to performing some initialisation. If you click on the 'Download Program' button in Visual BASIC, the program is downloaded to the memory of the RCX. The user must then select program block 1 using the grey button, and then start the program using the green 'Run' button.

Is it Visual BASIC or…

In the previous example, only direct commands were used in the RCX program. However, Spirit.OCX also recognises all sorts of control structures (such as Loop – End Loop, While – EndWhile and so on) that can be included in an RCX program. VB is actually nothing more than a shell, in which a program is recorded in a dialect resembling BASIC. This is why we earlier remarked that everything we have to say about VB can also be directly applied to Delphi and C++. The 'RCX dialect' is so similar to other programming languages that most of its functions and instructions do not need much explanation. For this reason, we will limit ourselves as much as possible to things that are specific to RCX.

RCX internals

From the perspective of the Lego software, the RCX is nothing more than an intelligent block with all sorts of attached sensors and motors, and there is no need to know everything that goes on inside that yellow-grey object. However, from the perspective of Visual BASIC it's a different story, since we have to deal with the internal timers and memory of the RCX. Consequently, we have to look at how to work with the RCX at this lower level.

Variables

Up to now, we have only used the functions of the RCX, but it also has room for storing data. It is even possible to store a whole series of measurements or other data, for later processing in the PC.

The RCX has 32 global variables (registers) that can contain values between –32,768 and 32,767. In principle, this means that all tasks and subroutines have access to these variables. This means that you have to be careful in the software not to unintentionally allow one task to modify a variable that is needed by another task. This can be most easily accomplished by giving each task exclusive access to certain registers, but this means that the tasks cannot exchange data. A more elegant solution is to use semaphores. A semaphore is a register (for example) that indicates whether a task is allowed access to the global variables, and if so which task. Other tasks must then wait until this task releases the registers. The semaphore register can for example have the value '0' when all registers are freely accessible. A task can reserve the registers by writing a unique value (which is assigned to the task) to the semaphore register. All other tasks can then access the registers only after the semaphore register once again

Table 3. Possible sensor modes.

Number	Constant	Sensor mode	Description
0	RAW_MODE	'raw'	data 0 – 1023
1	BOOL_MODE	boolean	TRUE or FALSE
2	TRANS_COUNT_MODE	transition counter	counts level changes
3	PERIOD_COUNT_MODE	period counter	counts complete periods (negative edge + positive edge)
4	PERCENT_MODE	percentage	sensor value as a percentage of full scale
5	CELSIUS_MODE	Celsius	temperature in degrees Celsius
6	FAHRENHEIT_MODE	Fahrenheit	temperature in degrees Fahrenheit
7	ANGLE_MODE	angular	rotation in terms of number of steps

has a value of '0'. This means that each task must release the registers when it no longer needs the global variables (for the time being). By the way, this difficulty should be eliminated in version 2.0 of the RIS firmware, which allows local variables to be used (the pre-alpha release of version 2.0 is available at the MindStorms site). However, since this possibility is (not yet) implemented in the Spirit.OCX module, we cannot make use of it in this context. However, Lego has added its own scripting language to the new version of the firmware, and this does incorporate all the new possibilities.

Any register can of course be read and written, using the Poll and SetVar methods respectively. Spirit.OCX also includes a collection of arithmetic and logical operators that can be applied to variables.

The RCX can also be used as a data logger. The values of timers, variables, sensors and the internal clock of the RCX can be logged. For each 'measurement value', the type, sequence number and value are stored. Table 1 lists the type and sequence numbers that are stored in the RCX memory for data logging.

The first thing you must do is to use the SetDatalog method to specify how many measurement values you want to store. This method returns a logical variable that indicates whether sufficient memory space is available. Normally, there should be enough room for around 2000 measurement values. However, it is always a good idea to check the logical value of SetDatalog using the following construction:

```
if
PBrickCtrl.SetDatalog(size)
then
   'enough room
else
   'too little room
```

If there is enough room for the block of data, a small black circle will appear at the right of the RCX display. This will be filled in during the logging session, one quadrant at a time, according to how many values have been stored in the memory.

After the SetDatalog method has been used, values can be stored during the logging session using DatalogNext. Finally, all the data logged during the session can be uploaded using the UploadDatalog function. The RCX sends the PC a three-dimensional array containing the previously mentioned values. If we for convenience assume that the array containing the values is named DatalogArray in VB, then the array element DatalogArray (0,0,2) holds the number of measurement values obtained during the session.

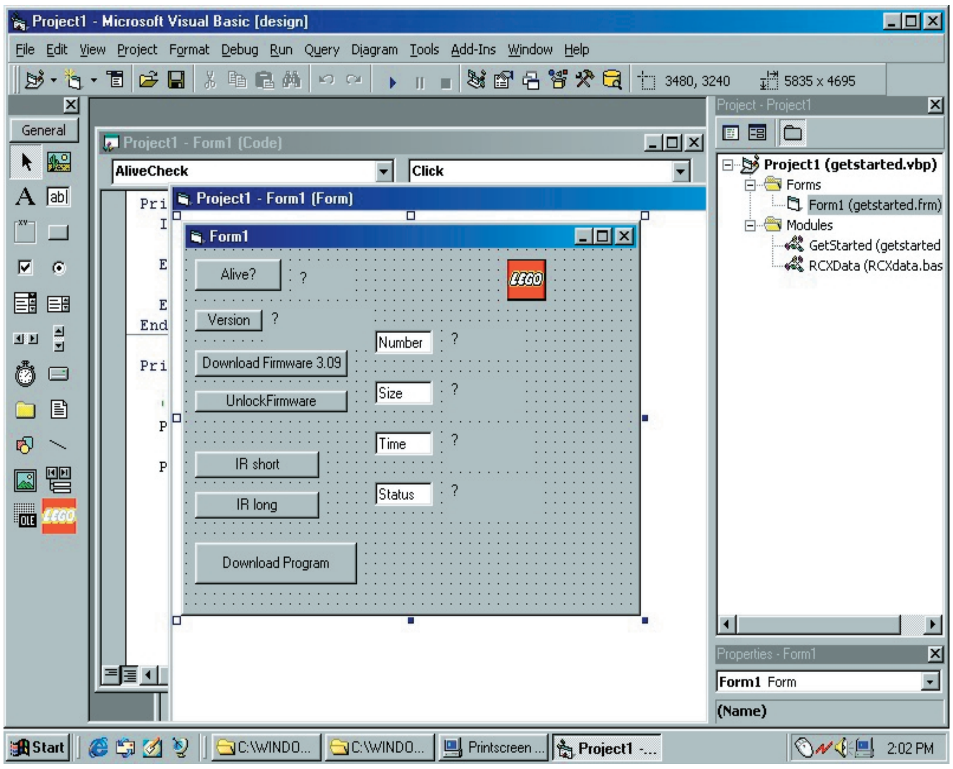


Figure 1. A sample application in Visual BASIC.

Motors

Various functions related to controlling the motors are available. The names of these methods are self-explanatory: On, Off, SetFwd, SetRwd, AlterDir and SetPower.

Sensors

In the Lego software, a block that belongs to the desired type of sensor can be selected, and this naturally must be declared in VB as well. For this, we use the function SetSensorType, which has two parameters. The first parameter is the number of the input to which the sensor is connected, and the second parameter is the sensor type. **Table 2** lists the possible types of sensors.

With the method SetSensorMode, we can specify the working mode of a sensor, or in other words, what values we expect to receive from the sensor (see **Table 3**).

The function Poll is used to read the value of a sensor, in the same way as it is used to read variables (registers).

Conclusion

In this instalment, we have seen how the RCX can be programmed using Visual BASIC. It clearly does not matter all that much which high-level language we use, since the actual control of the this oversized Lego block takes place via functions that are defined in Spirit.OCX.

In the next instalment of the series, we give our attention to communications between two RCX blocks, and we will work with a real robot.

(000040-3)