JANUARY 2001
£2.95

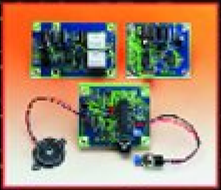
ELEKTOR ELECTRONICS

THE ELECTRONICS & COMPUTER MAGAZINE

www.elektor-electronics.co.uk

EPROM Emulator For the PC

High-Voltage Supply



Speed Controller Duet

Radio Linked Caller ID System

GameBoy Prototyping Board

µP Controlled Light Dimmer

MIDI to Voltage Converter

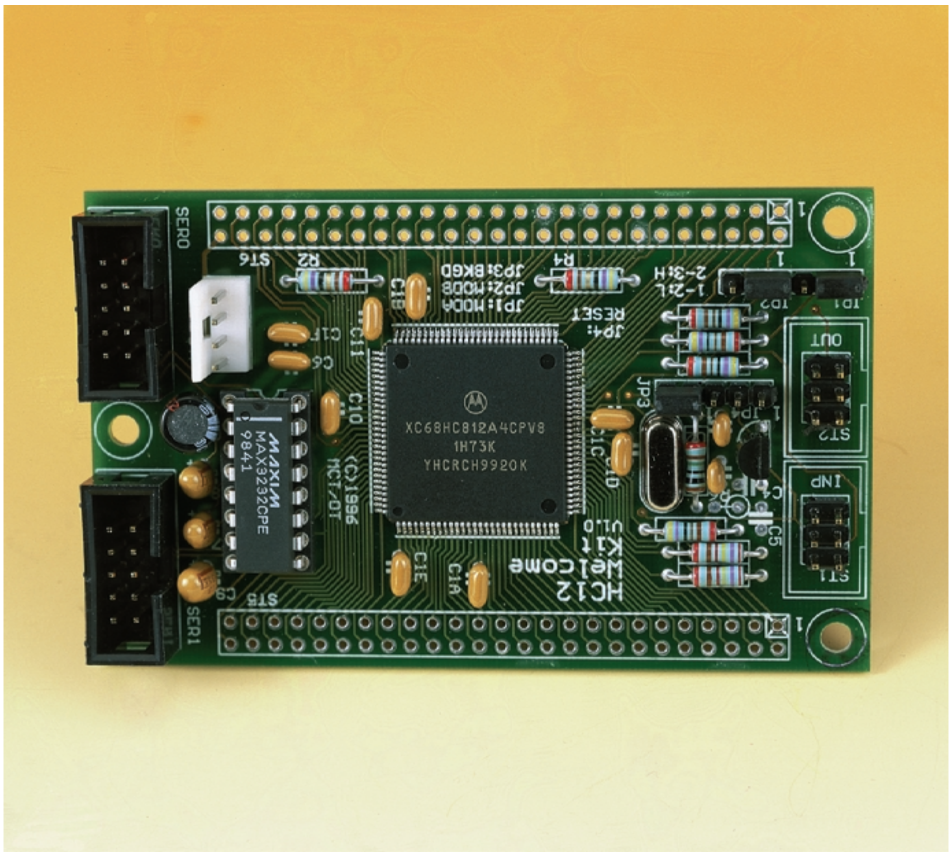
9 770268 451081 01

Upgrading to the 68HC12 16-bit microcontroller (1)

an introduction

By Oliver Thamm

The application of microcontrollers on the other side of the 8-bit boundary often seems like a closed book. Here we show that it can actually be fairly easy to start working with such microcontrollers — it's not even necessary to be an accomplished 'HC11 programmer!



Circuits using microcontrollers are very flexible, and they enjoy a correspondingly high level of popularity. The Motorola 68HC11 family has for years been one of the favourites for hobby applications. These 8-bit microcontrollers have an easily understood internal structure and can be readily programmed via an RS232 link. The HC11 fan club has grown over the years, and the number of published designs has also increased (see Literature). However, these days the HC11 has become somewhat technically obsolescent. New application areas demand controllers that are more flexible and that have more processing power. Motorola, the manufacturer of the HC11, finally decided to make a radical change in their manufacturing technology, which however could not easily be carried out with the HC11 types. The result was the 68HC812A4, the first member of the new, powerful HC12 family.

Many HC11 users do not realise that they can easily change over to

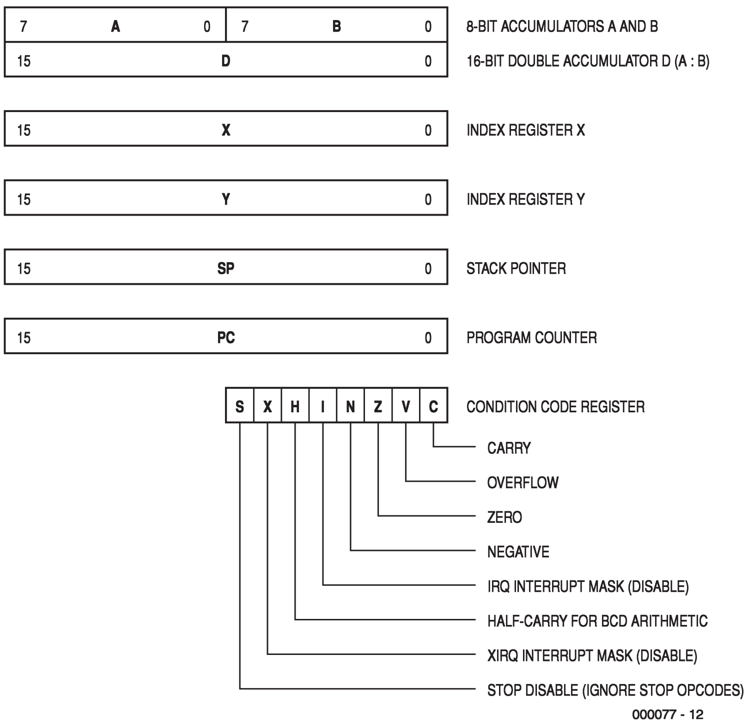

7 A 0 7 B 0
8-BIT ACCUMULATORS A AND B
15 D 0
16-BIT DOUBLE ACCUMULATOR D (A : B)
15 X 0
INDEX REGISTER X
15 Y 0
INDEX REGISTER Y
15 SP 0
STACK POINTER
15 PC 0
PROGRAM COUNTER
S X H I N Z V C
CONDITION CODE REGISTER
CARRY
OVERFLOW
ZERO
NEGATIVE
IRQ INTERRUPT MASK (DISABLE)
HALF-CARRY FOR BCD ARITHMETIC
XIRQ INTERRUPT MASK (DISABLE)
STOP DISABLE (IGNORE STOP OPCODES)
000077 - 12

Figure 1. HC12 CPU registers.

HC12 Operating Modes

The HC12 can work in several different operating modes that are selected via pins MODA and MODB. The levels on these pins are read in by the processor on the rising edge of the reset signal, to determine the operating mode. Normally, the BKGD pin must maintain a steady High level at this moment. The following operating modes, which are the most important, are frequently encountered in HC12 circuits.

Normal Single Chip Mode (MODA=L, MODB=L)
The HC12 operates without an external bus interface, using only the internal memory (RAM, EEPROM and Flash EEPROM if available). The bus leads that are thereby made free are made available as general-purpose I/O leads.

Normal Expanded Wide Mode (MODA=H, MODB=H)
The HC812A4 operates with a 16-bit-wide external bus interface. Ports A and B provide the address bus, while the data bus is found at ports C and D. Bus control is provided by the signals ECLK, R/W and /LSTRB (port E). The HC12 generates up to seven chip select signals on port F, as necessary. In addition, it is possible to use Memory Banking to drive memory regions beyond the 64-kB boundary. Up to six leads of port G are responsible for this.

Normal Expanded Narrow Mode (MODA=H, MODB=L)
In this mode, the HC812A4 behaves the same as in the Expanded Wide mode, except that the data bus is only eight bits wide (Port C). The MCU automatically splits every 16-bit access into two sequential 8-bit accesses.

Special Single Chip Mode (MODA=L, MODB=L)
This mode is specially provided for putting the IC into operation with the Background Debug Interface Mode interface. When the reset signal is released, the connected BCM pod must hold the BKGD lead Low for a short time. The HC12 system and the pod subsequently communicate via this lead. The CPU does not start with processing an application program, as usual, but initially executes only BDM instructions. In this software-development operating mode, the restrictions on the use of some of the control registers are relaxed, since some of the protective provisions of the MCU are disabled.

the faster HC12. This is because the new models have inherited the complete HC11 programming model. That means that the structure of the CPU registers (see **Figure 1**), the assembler instructions and the addressing modes have remained the same. Programmers will thus find it very easy to change to the new family, since existing source code only has to be retranslated. If you're only starting, rather than switching over, you can still enjoy certain advantages. In brief, the HC12 masters an important balancing act, combining high performance with a relatively low level of complexity.

Practical circuits

Enough of the introductory remarks, now it's time to look at a circuit diagram for the HC12. **Figure 2** shows an HC812A4 IC and the external circuitry that the microcontroller needs to run. Is that all it takes? Yes, indeed, and this is one of the main advantages of this microcontroller unit (MCU) – almost all the important components are integrated into the controller.

IC3 is the well-proven Maxim TTL/RS232 signal-level converter. The HC812A4 has two asynchronous serial interfaces, which are internally designated SCI0 and SCI1. Their operation is mutually independent, and they easily manage a data rate of 38,400 baud with the clock crystal used here. A second serial interface has been on the wish list of a lot of HC11 users for a long time. Now you finally have all the resources you need, for example to exchange messages with the application program via the first channel while sending debugging data via the second channel. The two connectors X2 and X3 are implemented as pin headers, to which one-to-one crimped flatcables can be connected, with 9-pin sub-D connectors (also crimped) at the PC end.

A quick tour of the MCU

The clock is generated using the usual combination of a quartz crystal, a pair of ceramic capacitors and a resistor. HC11 users will recognise the circuit, but in this case a 16-Mhz crystal provides the time standard. Compared to standard HC11 types, this provides four times the internal processing speed (8 MHz E clock).

In order to reliably reset the CPU when the circuit is switched on (or restarted), an external reset IC is prescribed. In our circuit, this service is provided by IC2, an MC34064P5. In addition, it is possible to connect a manual reset pushbutton via jumper JP3.

Jumpers JP4 and JP5 are normally closed

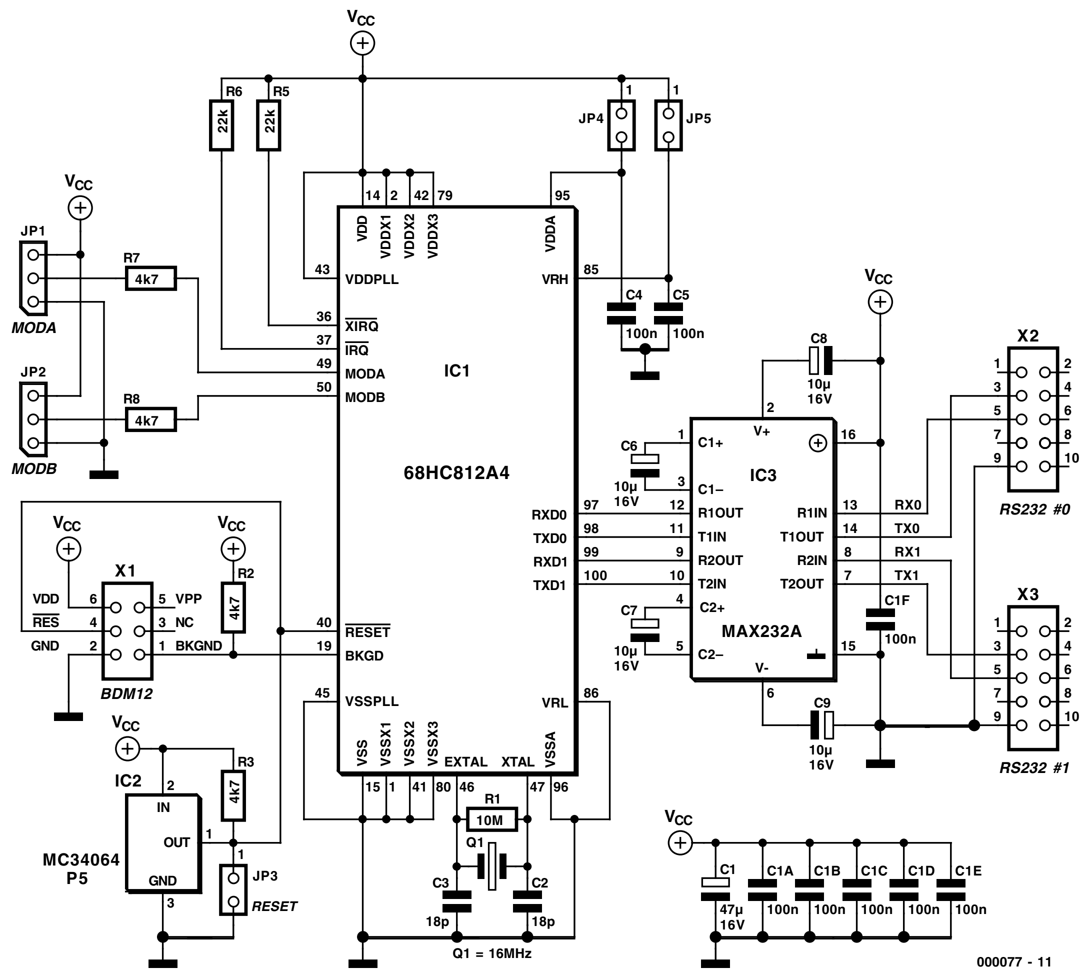


Figure 2. The circuit diagram of the HCA812A4 component set corresponds to that of the HC12 Welcome Kit.

(installed), so that the operating voltage (Vcc, +5 V) is applied to the power supply and reference voltage inputs of the integrated A/D converter. These inputs can be fed from external voltages by removing the jumpers, for example if it is necessary to use a more precise reference voltage.

Mode selection

Two additional jumpers are needed to set the operating mode of the HC12. JP1 is connected to the MODA pin, and JP2 to the MODB pin. The manner in which these signals affect the behaviour of the HC12 is explained in the 'HC12 Operating Modes' box. In our case, both jumpers are set to the 1–2 positions and thus select the Single Chip operating mode. Single Chip mode does not mean that there are not any other ICs on the board. Instead, in this mode the HC12 disables all bus signals for external memory ICs. The ports that are made free as a consequence can be used as general-purpose I/O lines, and this means a gain of up to 40 lines!

The internal EEPROM serves as the program store. With a size of 4 kB, it is larger than the internal program store of the HC11. In addition, there is 1 kB of RAM for variable data. Since the instruction set of the HC12 is very efficient, it is possible to store a fairly sizeable program in the internal memory. Some examples will be provided in the second part of this article, in the December 2000 issue.

In the background

Apart from a few blocking capacitors and pull-up resistors, we would be finished with the description of the schematic diagram, except for the six-pin connector X1. Only two 'real' signals are tied to this connector, namely /RES and BKGD. The signal /RES is an extension of the MCU /RESET signal, while BKGD is connected directly to the microcontroller. The built-in debugging function of the microcontroller works via this lead, which is the interface for the Background Debug Mode (BDM). As the name indicates, the BDM interface is used for more than just downloading software into the RAM or EEPROM. The BKGD pin also provides a 'back door' to the microcontroller, which allows any desired memory addresses to be read and written, ranging from the RAM and EEPROM to the IC control registers. This means that, via this pin and BDM, you can start and stop application programs as desired, set breakpoints, run software in single-step mode and read or modify the contents of the processor registers. The most important details of BDM12 operation are explained in the box 'How does BDM work?'.

This all sounds like we have a complete hardware emulator on board. Normally, you can expect to pay the cost of a small car for such a piece of equipment, but the HC12 includes the most important features of such a debugging aid at no extra charge. This sounds almost too good to be true, and in fact there is a fly in the ointment.

The connection to a PC that is used for developing and downloading the software is not entirely without problems. The BDM interface is simply too fast to work with a normal serial or parallel PC interface. This means that you need a bit of hardware and software to act as an interpreter between the PC and the HC12. This is called a BDM pod, and it is not particularly suitable for DIY construction. Fortunately, there are alternative solutions, such as the NoICE debugger, that will not soak up the annual budget of an ambitious HC12 developer in one blow. A large collection of links relating to BDM and the general subject of HC12 microcontrollers can be found on the author's HC12 Internet site at http://hc12web.de.

I/O leads

It's a fortunate person who has enough I/O heads to connect all of his LCDs, pushbuttons, lamps and relays. With an HC812A4, you won't find your self in a predicament quite so quickly, at least if it is working in Single Chip mode. More than 80 I/O connections are available to the user, each of which can be addressed as an input and/or output. With regard to the internal pull-up resistors, there are several configuration options. A substantial number of input signals (roughly one third) can function as interrupt triggers.

28 Pins × 4

If you're already getting curious and would like to take a closer look at the HC12, you have basically two options for getting started quickly. The first option is to obtain the microcontroller IC and mount it in a socket on a universal printed circuit board. The relatively few external connections will not present that much of a problem. However, the catch is that it is very difficult to manually solder the IC, so that a socket must be used – and suitable sockets for the 112-pin QFP package of the HC812A4 are expensive and hard to come by. In fact, they are so expensive that it is worth taking a look at the second option. This consists of a pre-assembled controller module, as shown in the photograph. It not only includes the peripheral circuitry shown in **Figure 2**, but also makes all processor signals available on lateral pin strips, where they can be easily accessed (even using non-professional means).
The German firm *Elektronikladen* (see http://www.elektronikladen.de on the Internet) offers such modules in credit-card format under the designation 'HC12 Welcome Kit' (see http://elektronikladen.de/kit12.html) The MCU is soldered directly to the circuit board in an SMD production step, without a socket. This makes the whole thing affordable, and for around £75 you receive a complete starter kit, including a monitor program for easy downloading of application programs via the serial interface.

(000077-1)

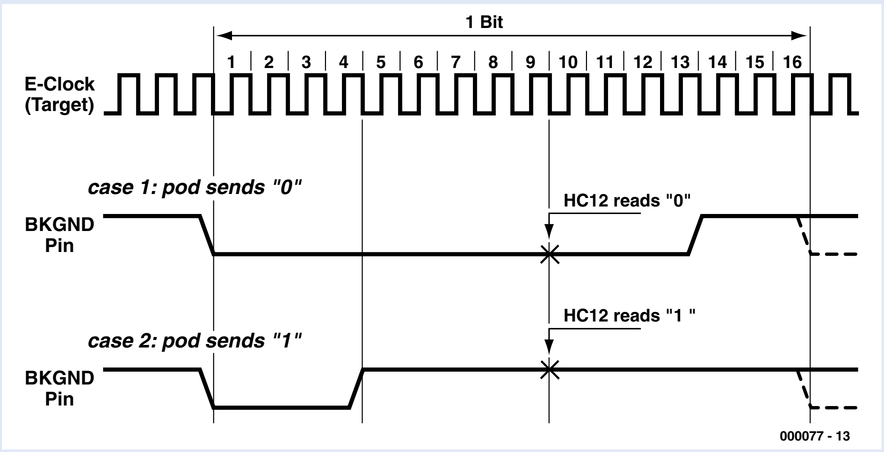
Part 2 will discuss the programming of the internal functional modules of the HC12 and the tools that are needed for this.

Literature:
68HC11 emulator,
Elektor Electronics February 1997.

How does BDM work?

The Background Debug Mode (BDM) interface of the HC12 is basically supported by a single lead (BKGD) of the microcontroller. In the idle state, this lead is held High by a pull-up resistor. A special asynchronous serial protocol is used for communications. The underlying clock is based on the clock speed of the HC12. For example, if a 16-MHz crystal is used, the basic time unit of the BCM system is 125 ns.
The timing diagram shows how an individual bit is transferred. This requires 16 clock cycles, which means that it takes 2 µs to transfer one bit at 125 ns per clock, and a full byte thus takes 16 µs. A typical BDM command sequence consists of five bytes, and thus takes around 80 µs. In practice, you can expect to see around 5 to 10 thousand BDM instructions per second – and that's already pretty fast.
A BDM instruction consists of a command byte and any necessary parameters. For example, the instruction sequence for a 16-bit write access to address $0800/1 appears as follows:

```
C8  08  00  1A  2F
```



The individual components of the message have the following meanings:

```
$C8    BDM command 'WRITE_WORD'
$0800  address
$1A2F  data word
```

Learning RC5 Control Decoder

build a low-cost remote volume control

By Jason Vincent-Newson

jasreb@bigfoot.com

The unit described is intended to be used with any Philips RC5 remote control to enable up/down remote control of a motor driven potentiometer.

A special feature of the design is that it is truly universal in that it can be programmed to respond to any two specific keys on any Philips RC5 compatible remote control.

This is achieved simply by shorting two pins on the unit during power-up. The unit will then indicate PROG mode by flashing the relevant LEDs, and learn the system address and volume UP/DOWN commands as they are pressed on the chosen remote control. The resulting information is stored in non-volatile memory for retention after power-down.

The unit can be reprogrammed at any time in the same way.

A power opamp motor driver is included to directly drive any conventional 5-V motor driven pot.

The PCB is fitted with a 10-way pinheader connector to allow easy design into new or existing equipment. The pins give access to the 3-pin IR sensor, motor wires and the PROG jumper so that these can be sited remotely if required.

Circuit Description

As you can see in **Figure 1**, the circuit is very simple and consists of three sections: an infrared (IR) receiver/demodulator type IS1U60 (IC3), an 8-pin AVR microcontroller AT90S2343 (IC1) and a power opamp

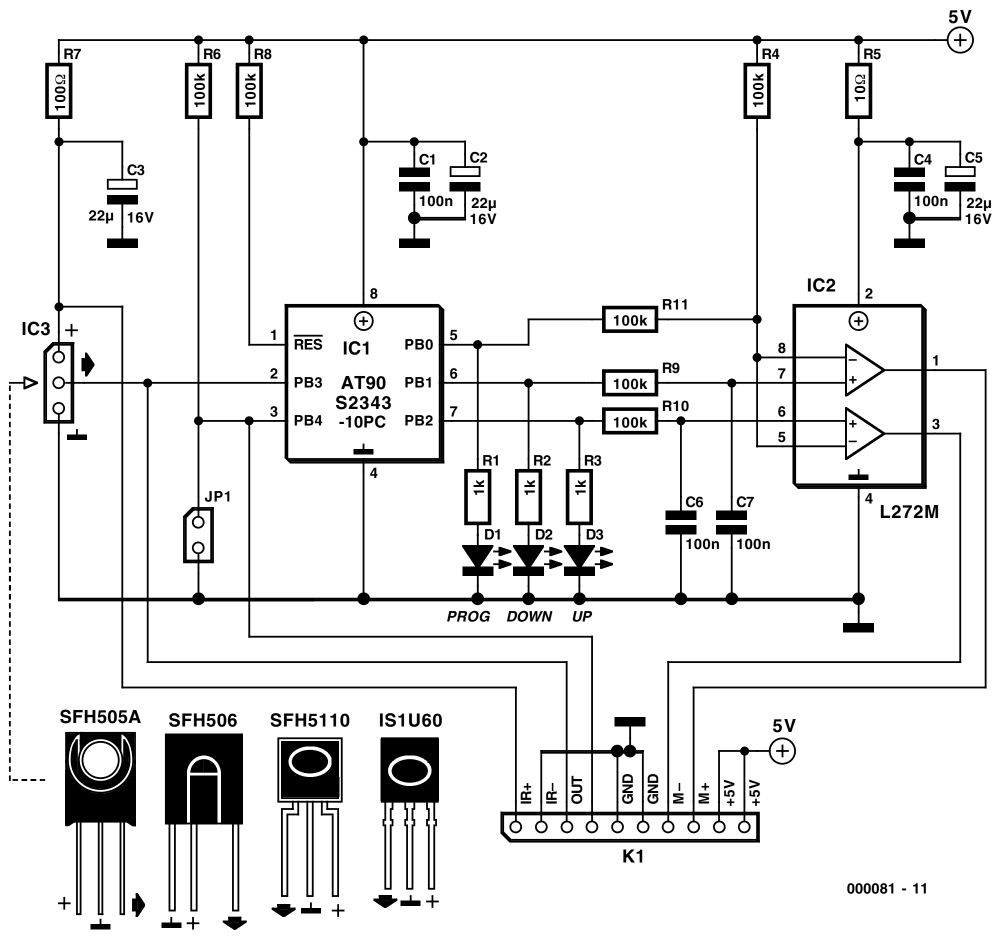


Figure 1. Circuit diagram of the Learning RC5 Decoder. The L272M power opamp drives a 5-V stepper motor.

motor driver L272(IC2).

The IR receiver/demodulator receives the modulated RC5 data stream and provides a demodulated 5-V digital output for processing by the microcontroller. If you can not get hold of the Sharp IS1U60, the Siemens near-equivalents SFH505A, SFH506 or SFH5110 may be used instead. These IR decoders are not pin compatible hence their pinouts are shown in the circuit diagram. Although the SFH505A is now 'obsolete' according to Siemens, you may still manage to get your hands on one.

The microcontroller used is an AVR AT90S2343 8-pin device. This tiny device includes 2k of Flash program memory and 128 bytes of non-volatile EEPROM user memory. An internal oscillator runs at 1 MHz and the efficient design of the processor enables processing approaching 1 MIPs. The microcontroller is available ready-programmed through our Readers Services.

The processor has five I/O pins, all of which are used in the present design:

- PB3 is used as the RC5 data input, active Low.
- PB4 reads the PROG jumper (PROG mode selected when Low).
- PB1 and PB2 are the UP/DOWN outputs to drive the L272 motor driver (the UP/DOWN indicator LEDs are also driven via these pins).
- PB0 doubles as the motor enable output (when Low) and the PROG mode indicator (when High).

The L272 motor driver operates in the following way:

The two –inputs are tied together and connected to a potential divider R4-R11 that sets the voltage of these pins to approx. 2.5 V when the enable output (PB0) is Low. In this mode, taking input pin 7 High and the corresponding input pin 6 Low will drive the separate outputs of the L272M High and Low respectively. Reversing the input pins (6 High, 7 Low) will reverse the outputs hence reversing the direction of the motor.

If the PROG output (PB0) is HIGH the opamp is disabled because the –pins (8 and 5) are pulled to +5 V, making it impossible to turn On either half of the opamp.

The 10-Ω and 100-Ω resistors and the

DIY AT90S Microcontroller

Those of you with access to an AT90S programmer will be pleased to know that the source code (Bascom AVR Basic) and the Hex file for the microcontroller used in this project are available from the Free Downloads section of our website at http://www.elektor-electronics.co.uk. Look for file # **000081-11.exe** under the December 2000 items.

corresponding supply filter capacitors serve to filter out any supply noise present on the 5-V rail.

The quiescent supply current of the unit is negligible and the operating current is determined by the current drawn by the motor used.

Construction

The unit is constructed on a small single-sided PCB as shown in **Figure 2**. IC2 may be either soldered directly or fitted in a DIL socket as preferred, but the microcontroller (IC1) should be fitted in a socket.

If it is intended to use the board as a standalone control, a straight pin header can be fitted in position K1 and the power supply and motor connections made directly to the pins. The IR sensor may be connected with flying wires to enable it to be sited remotely.

If it is intended to incorporate the unit into an existing piece of equipment then the pin header can be plugged into a suitable female connector (socket).

Operation

The unit is extremely easy to set up and operate. On power-up the unit will first check the PROG jumper and if set will enter PROG mode:

PROG mode
The red (PROG) LED will light and the Down LED (D2) will flash to indicate that the Volume Down key on the remote must be pressed. Once the command (and remote control system address) have been correctly received the Down LED will stop flashing and the UP LED with start to flash. The Volume Up key is now pressed on the remote and once correctly received the red PROG LED will go out and the unit will enter NORMAL mode:

NORMAL mode
In this mode the unit continuously monitors the IR sensor and if it detects the programmed volume Up/Down commands will drive the motor (+/–) outputs in the appropriate direction. If the key is held down the motor will continue to turn until it is released.

On power-up (assuming the PROG jumper is not set) the unit will look to see if there is valid data stored in the non-volatile memory for the stored commands. If none is found (i.e. the first time the unit is powered up, or if the data has been lost for some reason) the unit will automatically enter PROG mode as described above.

(000081-1)

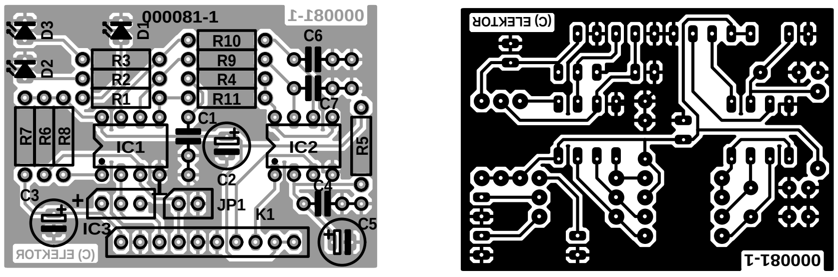


Figure 2. Copper track layout and component mounting plan of the small PCB designed for the control (board available ready-made through Readers Services).

COMPONENTS LIST

Resistors:
R1,R2,R3 = 1kΩ
R4,R6,R8-R11 = 100kΩ
R5 = 10Ω
R7 = 100Ω

Capacitors:
C1,C4,C6,C7 = 100nF
C2,C3,C5 = 22µF 16V radial

Semiconductors:
D1,D2,D3 = low current LED
IC1 = AT90S2343-10PC, Publishers order code: **000081-41**
IC2 = L272M

Miscellaneous:
JP1 = 2-pin pinheader with jumper.
K1 = 10-way SIL pin header for board edge mounting.
IC3 (not on board) 36 kHz centre frequency IR receiver/demodulator, e.g., IS1U60 (Sharp) SFH506-36, SFH5110-36, SFH5110 (Siemens).
PCB, Publishers order code **000081-1**.
Disk, project source code and Hex file, order code 000081-11

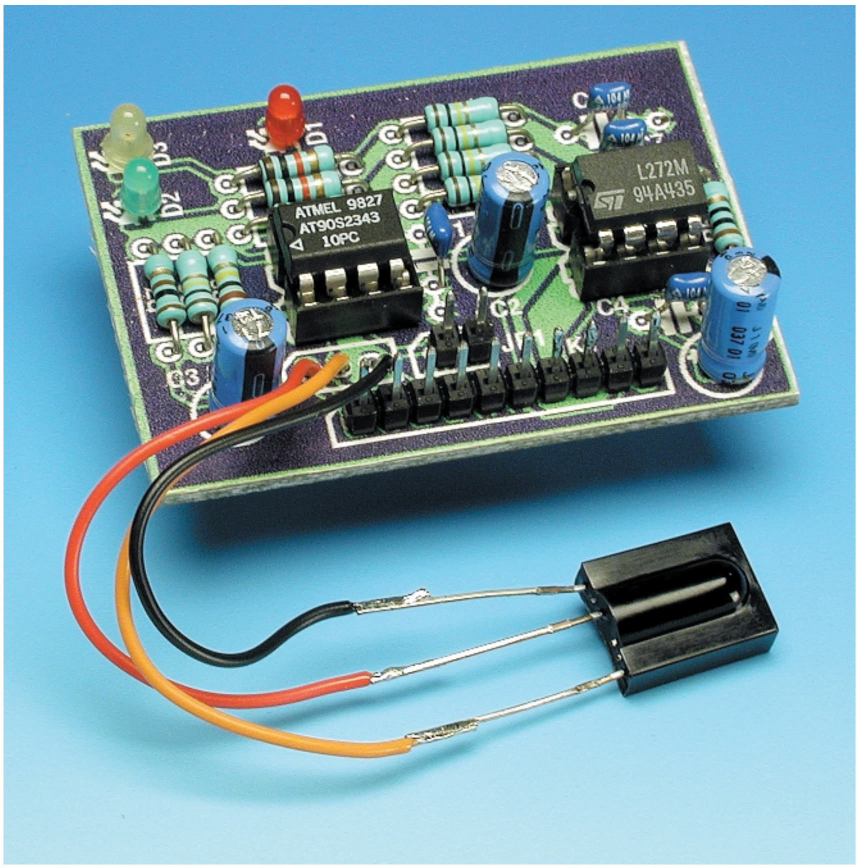


Figure 3. Compare your work against this working prototype of the Learning RC5 Decoder.

Universal Mobile Telephone Service

Part 2: Third generation mobile technology

By G. Kleine

The introduction of Third Generation mobile phones will offer vastly increased data capacity ensuring the proliferation of mobile Internet use and new multimedia uses.

To ensure that global roaming would be possible for third generation mobile phones the World Radiocommunications Conference (WRC) in 1992 defined the operating frequencies for UMTS to be 1885 MHz to 2025 MHz (uplink) and 2110 MHz to 2200 MHz (downlink). Uplink is defined as information passing from the subscriber to the base station while downlink is information passing from the base station to the subscriber. **Figure 1** shows UMTS frequency allocation. The upper band of these frequencies is reserved for future Mobile Satellite Service (MSS). The core frequencies for Frequency Division Duplexing (FDD) with W-CDMA modulation are 1920 MHz to 1980 MHz for the uplink and 2110 MHz to 2170 MHz for the downlink. These maintain a duplex distance of 190 MHz between the uplink and downlink. FDD is favoured by Nokia and Ericsson and provides identical capacity for the uplink and downlink channels. Current GSM systems use FDD with respect to the uplink and downlink capacity. An alternative to FDD is the Time Division Duplexing (TDD) System that has been standardised by Bosch, Siemens and Alcatel. A major advantage of this standard is the use of non-paired frequency bands thereby allowing unsymmetrical channel capacity between the uplink and downlink channels. This is much more suited to the ‘bursty’ nature of TCP/IP data exchange that occurs when accessing Internet information. In this case uplink data could consist of little more than a web page address which results in a downlink data flow of many pages of data. For FDD a Duplex bandpass filter with relatively sharp cut off characteristics is used to prevent coupling of send and receive data. With TDD equipment it is necessary to strictly control the synchronisation between the base stations so that adjacent cells can function correctly together and that the entire network can operate synchronously.

The two UMTS frequency ranges each 60 MHz wide split into 12 channels each with a 5 MHz bandwidth are at the moment available for use in European countries and in the majority of Asian countries. In the USA the situation is not quite so clear. The USA currently uses the

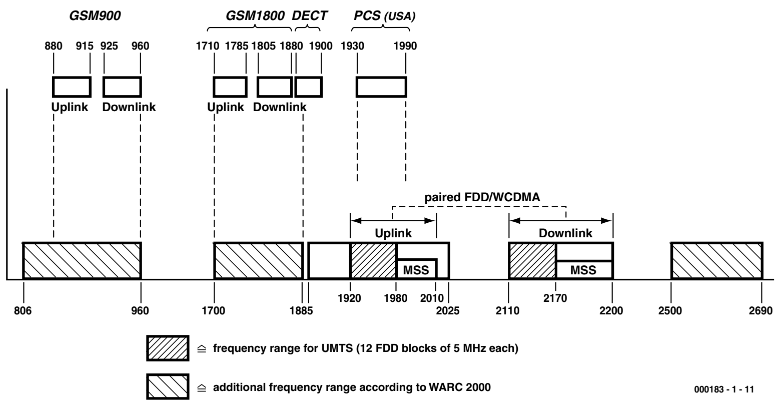


Figure 1.UMTS frequency allocation.

1.9 GHz band for its second generation phone network called Personal Communications System or PCS. The plan here is to free-up some of these frequencies for UMTS use in the coming year.

It is also anticipated by the WRC-2000 that the 806 MHz to 960 MHz band will be used for UMTS when the current GSM era comes to a close. The 1710 MHz to 1885 MHz will also become available when the DCS 1800 service comes to an end. Apart from these frequencies there is also a part of the spectrum which is currently used by DECT cordless phones. Lastly 2500 MHz to 2690 MHz is also eventually anticipated to be used for future capacity increase of the UMTS network in.

UMTS frequency licences

The allocation of these highly desirable 5 MHz frquency pairs has been handled differently in different countries. In Spain, Norway and Sweden for example, these licences have been distributed on a practically cost free basis whereas in most other countries they have been auctioned off to the highest bidder. Here in the U.K. five licences were available and four of these went to the big hitters in the mobile phone world: Vodafone, BT3G, one2one (owned by Deutsch Telecom) and Orange (owned by Vodafone but currently on the transfer list). The fifth licence was awarded to TIW, a relative newcomer comprising Canada Telesystem International and Hutchison Whampoa of Hong Kong. The U.K. auction for 3G licences raised £22.48 Billion for the Government.

More details of the licencing can be found in UMTS forums on the Internet.

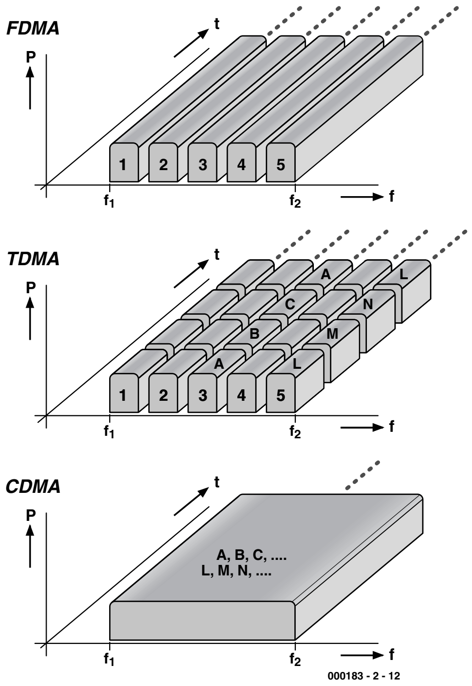


Figure 2. FDMA, TDMA and CDMA access methods.

A refined modulation method

The classical methods of providing multiple communication channels by using FDMA and TDMA were described in part 1 of this article in the previous issue of *Elektor Electronics.* **Figure 2** shows diagrammatically how the channel is subdivided in frequency and time when either FDMA or TDMA is used. Another method that we have not looked at yet is also shown. UMTS will use a Code Division Multiple Access (CDMA) method. CDMA allows all users to send simultaneously using the same frequency. At first glance this may seem a very poor method: If everyone is using the same frequency how is it possible to select just one transmitted signal at the receiver? A simple filter would attenuate all channels equally because they are all sending on the same frequency. The key to this method is that each transmit and receive channel is assigned a unique code or 'Pseudo Noise' (PN) sequence. Transmitted data is combined with this sequence before transmission. At the receiver, we detect what just seems like noise from these many transmitters all sending together however when this received 'noise' is passed through a correlator which uses the same code sequence as the transmitter, the original transmitted signal is magically recovered.

This concept may be difficult to grasp but as an analogy imagine that you are having a conversation with a friend in a lift. The lift stops and two people enter conversing in Japanese, next two people enter conversing in Russian. If we assume (crucially) that everyone in the lift can only understand their own language then you can see that these three conversations can occur simultaneously without any mix-up of information. Somewhere deep in your brain the equivalence of correlating the sounds at your ears with the code sequences of the English language is occurring.

The process of spreading can be seen as sacrificing a narrow bandwidth signal for a signal that can operate in a high noise environment and is more immune to noise. A simplified example is given here in **Figure 3**.

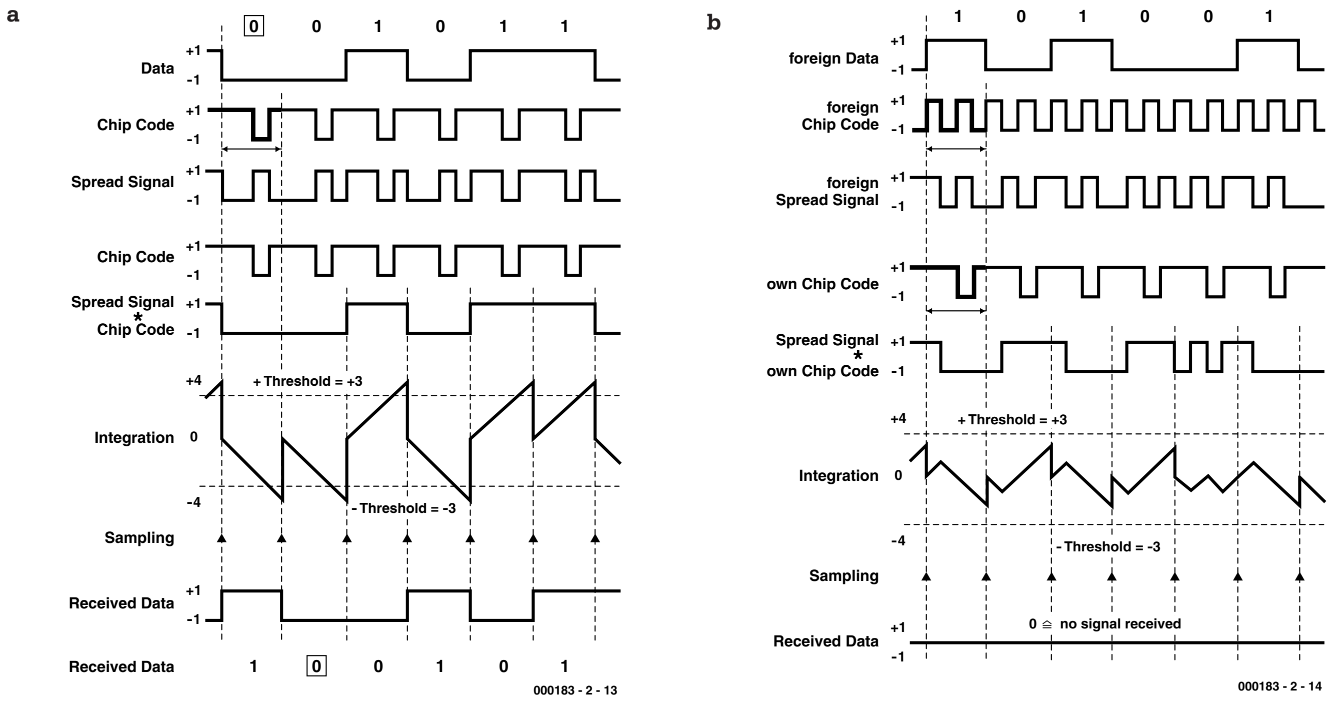


Figure 3. The principle of CDMA communication. a) Reception of the desired signal b) Reception of an unwanted signal.

Encoding of the transmission signal and reception of the desired signal with the correct Chip sequence is shown in **Figure 3a**. The Chip sequence data rate is four times the actual data rate in this example. Processing gain is a measure of by how much the bandwidth of the data signal is increased by the spreading process. It is defined as the ratio of the chip rate to the data rate. Each input data bit is multiplied by a four bit long chip sequence. A high is +1, a low is –1 and no signal is 0. The resultant transmitted data signal after multiplication with the spreading signal has a bandwidth of four times the data rate.

By signal reception the input data integrator is initially at set to zero. The integrator amplitude is given by the processing gain for the system. The sample time of the waveform occurs at the end of each integration period. The sampling threshold is here set to +3 and –3. At the output of this process we have recovered the actual data stream which was input to the transmitter with an additional time delay.

Figure 3b shows the same process but this time with an unwanted signal with an unknown chip sequence. The first two lines show the unwanted data signal and its chip sequence. Line three is the spread signal. As before line four is the chip sequence used by the receiver here we are using the same chip sequence as in **Figure 3a**. The next line shows the product of the above two lines and this is input to the integrator. We can see that at no point during the integration process does the signal step above or below the +3 and –3 threshold so the output remains at zero. Zero indicates that no signal has been detected. From this we can see that the chip sequence should be carefully chosen so that it has a sharp auto-correlation peak i.e. that it does not allow any other unwanted code to be mistaken for the desired code. Cross-correlation between two chosen chip sequences should be as small as possible (optimally zero).

The Wideband variant

Originally CDMA was used in a military environment and employed a relatively narrow signal bandwidth of 100 kHz. UMTS signals require a bandwidth of 5 MHz in order to support a data rate of 2 Mbit/s and a chip rate of 4,096 Mbit/s. The correspondance of the chip rate to the data rate is relatively small (2 Mbit/s or 2 MHz) yielding a relatively small processing gain. This low processing gain means that when several subscribers are sending on the same frequency it is crucial that each received signal must reach the receiver with approximately equal signal strength in order that the integration process at the receiver functions correctly (see The near/far problem of W-CDMA). Good quality receivers must be employed in the base stations and in the mobile handsets along with transmitter stages with good linearity otherwise intermodulation products will have adverse effects on the signal quality.

A W-CDMA communication system

A simplified block diagram of a W-CDMA communications system is shown in **Figure 4.** This example uses a data stream of 1,024 Mbits/s and is passed through an encoder and interleaver. The interleaver provides error protection by shifting bit positions so that the effects of interference bursts are distributed throughout the data stream and can be corrected once the signal is de-interleaved at the receiver. The resultant data stream is now multiplied by the chip sequence (12.288 MB/s in this example). This processing is equivalent to performing an XOR operation between the data and the spreading code (also known as 'chipping'). A bandpass filter limits the signal bandwidth to that of the chip rate. The original bandwidth of 1,024 MHz is spread to 12,288 MHz and the processing gain is 12.

The signal is now passed through the power amp (PA), to the antenna. As the signal travels through the ether to the receiving antenna it is subject to interference and noise. It also mixes with transmissions from other mobiles using the same frequency. The resulting noise-like signal is picked up by the receiver antenna, amplified by the input amplifier (LNA) and filtered through the chip rate bandwidth filter. The signal is now passed to the correlator this will mix the signal with a locally generated chip sequence that must be synchronised with the transmitter chip sequence. At the output of the correlator the actual transmitted data stream is reconstituted. Now after the de-interleaving and decoding the DATA OUT signal will correspond to the original DATA IN signal at the input.

Figure 5 shows the comparison between the present day radio cell structure compared to the radio cells that will be used in the W-CDMA system. Uplink and downlink capacity can be adjusted according to needs.

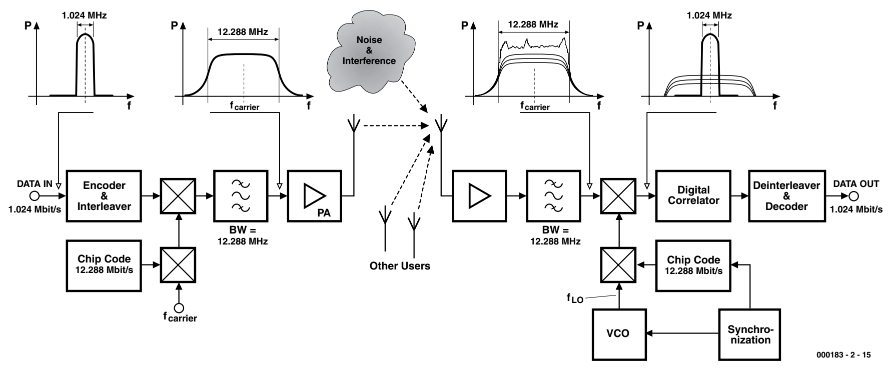


Figure 4. W-CDMA Communication system.

The near/far problem of W-CDMA

The principle that many subscribers in the same cell can use the same frequency relies on the necessity that each signal arriving at the base station must have approximately equal signal strength so that the decorrelation process can function correctly. This applies to the mobile 5 km away from the base station as well as the user just across the road from the base station. This is the so called near/far problem and going back to our analogy, it is equivalent to some people in the lift shouting, this is sure to upset everyone else's ability to carry on their conversation.

Unwanted signals cannot be removed with filters because they are using the same frequency as the desired signal. The method used in W-CDMA is for the base station to dynamically control the power output of each mobile so that the received signal strength at the base station from each mobile is equal. The output power of a UMTS mobile will need to be adjustable over a range of 70 dB i.e. 1:10 million! The power amplifier must also be capable of a switching speed of 1500 gain steps per second. In contrast current GSM mobiles only need to alter power output a few steps per second.

The future looks bright

Many companies are currently co-operating to develop UMTS base stations and prototypes of UMTS handsets. Key players in the production of net equipment and base stations are Ericsson (Sweden) and Nokia (Finland). Both companies have collaborated closely with research facilities and institutions to produce the first prototype of a UMTS system. Nokia has also won many contracts in China and Ericsson is collaborating with Vodafone to provide pilot implementation of a UMTS system in the UK.

Alcatel and Fujitsu are jointly working on development of GPRS, EDGE and UMTS technology and are in the process of promoting this on a bus tour of Europe demonstrating the principles and possibilities of new UMTS technology. Many companies are already producing dedicated UMTS chipsets. Infineon Technologies of Austria have a purpose built UMTS development centre This facility goes under the name of Danube Integrated Circuit Engineering (DICE) and it has already in April 2000 delivered the first UMTS chip for the Japanese market. Toshiba Corporation have also set up a small Telecommunications Research Laboratory in Bristol with the aim of developing UMTS chipsets.

As for the implementation timescale for UMTS it is anticipated that by 2001 we can expect the first field trials of a UMTS network and by the end of 2002 the system should be partially implemented leading to total coverage by 2005.

By 2010 GSM will probably still have many users, especially using GPRS and EDGE enhancements.

(000183-2)

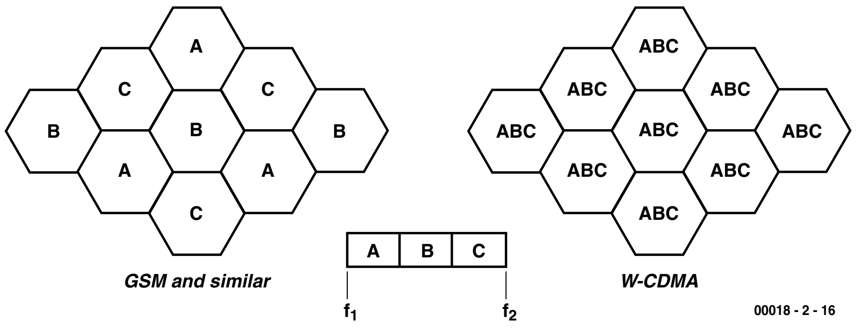


Figure 5. Radio cells and frequency use

PC Serial Peripheral Design (5)

Analogue measurements

By B. Kainka

So far in this series we have used the PC only with digital signals: switching, monitoring and counting. Now we turn to the analogue domain: our programs will understand not just 'yes' and 'no', but 'larger' and 'smaller'.

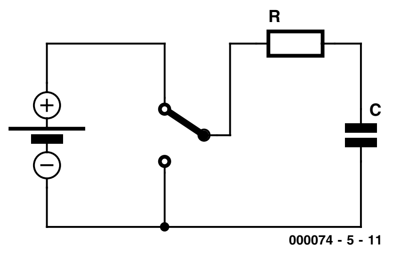


Figure 1. Charging and discharging a capacitor

If a voltage is applied to a serial port input, it will be read as either a logic Low (0) or a logic High (1). The PC cannot determine the actual voltage present. Likewise, if a resistor is connected between an output that has been set high and an input, there are only two possible results: either the resistor is sufficiently small that a clear logic one is read at the input, or it is not. If more precision is required, a new approach is needed.

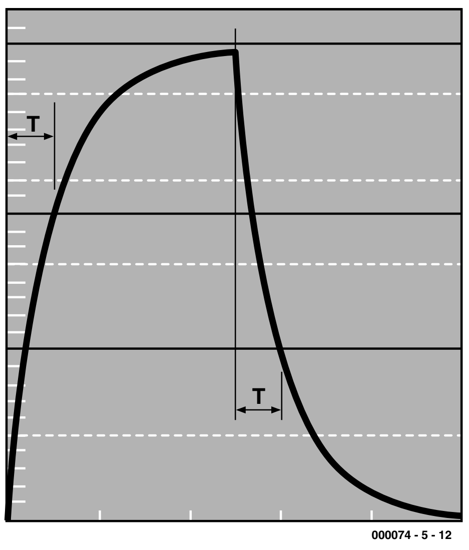


Figure 2. Charge/discharge curves and definition of the time constant T

Charging and discharging

If computers are good at one thing, it is counting. We can use this to measure time: a program simply counts the seconds (or milliseconds) that pass until a certain event happens, for example when an input changes state. If we can convert an analogue quantity such as a resistance into a time period, then it will be easy to measure it with a computer. In this example, we can use an RC network. **Figure 1** shows a capacitor C charged and discharged through a resistor R. A time constant *T* is associated with an RC network:

T = RC.

Here *T* is the time taken for the voltage across the capacitor to reach 63.2 % (=1-1/e) of its final value. This can be derived from the exponential charging characteristic, shown in **Figure 2**. We shall not go into the details of the physics here: it is enough to know that the time taken to charge to a given voltage is directly proportional to the capacitance and to the value of the resistor. For a 100 μF capacitor and a 1 kΩ resistor we have:

$T = 1000\ \Omega \times 0.0001\ \text{F} = 0.1\ \text{s} =$ 100 ms

A doubling of the resistance leads to a doubling of the charging time. The same goes for a doubling of the capacitance. So, we can measure the charging time and deduce the value of either one of the resistor or the capacitor, if the value of the other is known. All that is needed is a program that replaces the switch in Figure 1. **Figure 3** shows a simple circuit where the capacitor is connected not to ground but to the TXD output. There is a good reason for this: if an electrolytic capacitor is used, it must never be charged with the wrong polarity, and this is guaranteed as long as TXD remains at –10 V.

The circuit charges and discharges the capacitor via DTR and uses the DSR signal as an input to determine when the set voltage is reached. The threshold voltage will be around 1.5 V. Comparing this with the overall voltage range of –10 V to +10 V, we see that the threshold is about 11.5 V/20 V = 0.575 = 57.5 % of the final voltage. This is not too far from our value of 63.2 % given above for the time constant. In any case, the error factor introduced is constant and can be compensated for later. There are other sources of error, which we discuss below.

Listing 1. Measuring the time constant in milliseconds

```
Private Sub Form_Load()
 i = OPENCOM("COM2,1200,N,8,1")
 If i = 0 Then
    i = OPENCOM("COM1,1200,N,8,1")
    Option1.Value = True
 End If
 If i = 0 Then MsgBox ("COM Interface Error")
 TXD 0
 RTS 0
 DTR 0
 Counter1 = 0
 Timer1.Interval = 2000
End Sub

Private Sub Form_Unload(Cancel As Integer)
  CLOSECOM
End Sub

Private Sub Timer1_Timer()
  DTR 1
  TIMEINIT
  While (DSR() = 0) And (TIMEREAD() < 1501)
    DoEvents
  Wend
    Label1.Caption = Str$(TIMEREAD()) + " ms"
  DTR 0
End Sub
```

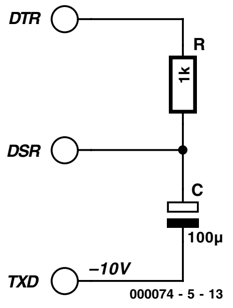
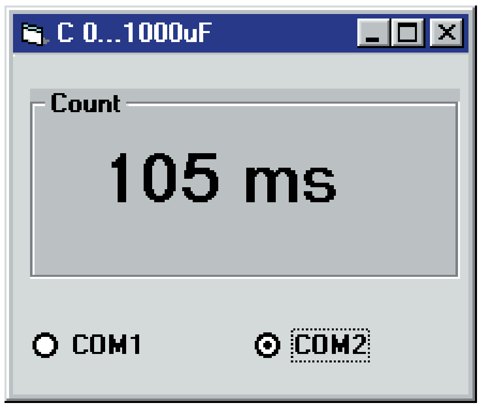


Figure 3. Automatic charging from the PC



105 ms

Figure 4. Measurement with 100 μF and 1 kΩ

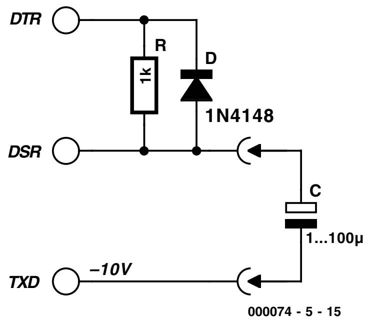


Figure 5. Improved capacitance measurement circuit

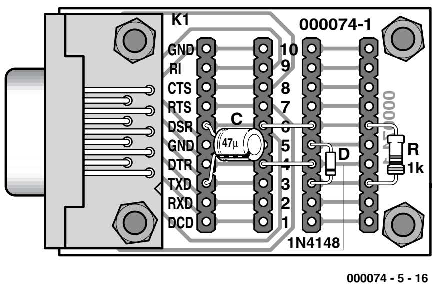


Figure 6. Construction details.

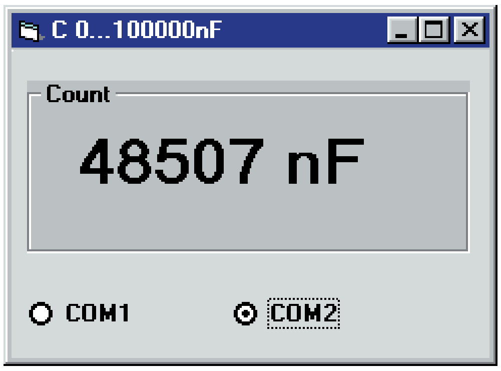
Counting time

The time measurement takes place in a 'while' loop as shown in **Listing 1**. The loop runs until either DSR becomes set or the count reaches 1.5 seconds (timeout). The measurement loop includes the command DoEvents: this allows Windows to process other events that may be occurring in the system. During measurements the user can still use the mouse and other applications, and indeed stop the program itself. This is reassuring for the user, especially when bugs in the program cause it to function incorrectly. In general it is always necessary, when programming loops, to consider how the loop can be forced to terminate. Otherwise if the program gets into an infinite loop the PC will need to be reset, either by switching it off and on again, or by using the familiar Ctrl-Alt-Del chord. Adding a DoEvents call makes the loop safe; but this has a cost in timing accuracy, adding an uncertainty of around 1 to 3 milliseconds to the measured time.

μF not ms

If the units 'ms' in the window in **Figure 4** are replaced by 'μF', the value shown is not too far off the true one. As we said above, a 100 μF capacitor and a 1 kΩ resistor give a time constant of 100 ms. Similarly, with a 10 μF capacitor we have a time constant of 10 ms. This can be tested by trying various capacitors from the junk box. Often there will be quite a large discrepancy between the value measured and the value printed on the capacitor: this is down to the large tolerance (as much as 50 %) quoted for electrolytics. The capacitance of an electrolytic often changes if it is stored for a long time.

The measurements are less accurate for very large electrolytic capacitors, with values of say around 1000 μF. The indicated value will be too small. The reason for this lies in the program: the charged capacitor must be discharged, which also requires time. Our program uses a timer with a period of two seconds: one second is used for charging the capacitor up to the input threshold voltage; the remaining second, however, is not enough to discharge the capacitor completely. The next charge therefore takes less time. There is a simple solution: a diode can be used to accelerate the discharge cycle. **Figure 5** shows the circuit diagram, and **Figure 6** shows how it can be constructed. With this modification to the circuit we can



C 0...100000nF

Count

48507 nF

( ) COM1 (●) COM2

Figure 7. Capacitance display in nF

Listing 2. Modifications for program Cmeas2.frm

```
Private Sub Timer1_Timer()
  F = 1.19
  DTR 1
  REALTIME (True)
  TIMEINITUS
  While (DSR() = 0) And (TIMEREADUS() < 1500000)
  Wend
  T = TIMEREADUS() * F
  REALTIME (False)
  T = Int(T)
  Label1.Caption = Str$(T) + " nF"
  DTR 0
End Sub
```

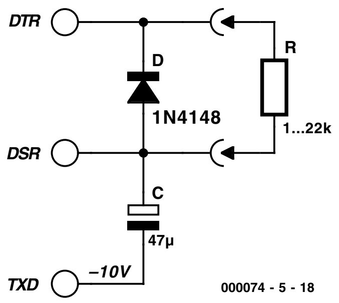


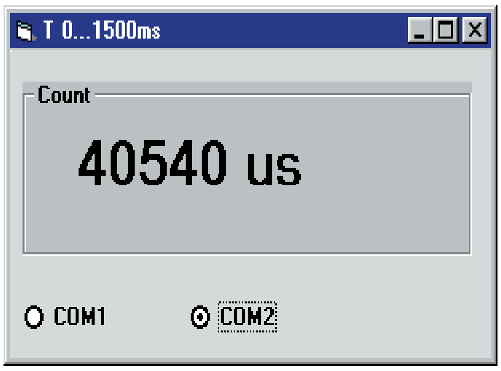
Figure 8. Circuit 1 with 10 kΩ potentiometer.

Listing 3. Measuring the time constant in microseconds

```
Private Sub Timer1_Timer()
DTR 1
  REALTIME (True)
  TIMEINITUS
  While (DSR() = 0) And (TIMEREADUS() < 1500000)
  Wend
  t = TIMEREADUS()
  REALTIME (False)
  Label1.Caption = Str$(t) + " us"
  DTR 0
End Sub
```

Listing 4. Determining the charging resistance

```
Private Sub Timer1_Timer()
  DTR 1
  REALTIME (True)
  TIMEINITUS
  While (DSR() = 0) And (TIMEREADUS() < 1500000)
  Wend
  T = TIMEREADUS()
  T = T * 1.0000000001
  R = 2200 + 7800 * (T - 76300) / (294600 - 76300)
  REALTIME (False)
  R = Int(R)
  Label1.Caption = Str$(R) + " Ohm"
  DTR 0
End Sub
```



T 0...1500ms

Count

40540 us

( ) COM1 (●) COM2

Figure 9. Measuring the charging time to microsecond resolution

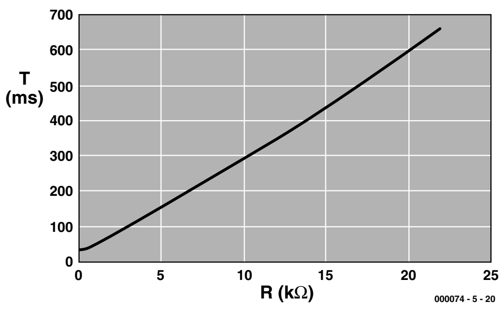


Figure 10. Charging time plotted against charging resistance

reliably measure capacitors up to about 1500 μF. Larger values are possible if more time is allowed by making the appropriate changes to the program. The timeout value in the measurement routine and the overall timer period must both be increased, so that the program waits long enough for the measurement to be completed.

What about capacitors smaller than 1 μF? In principle the charging resistor could be increased. However, a problem then arises: the impedance of the DSP input (around 3 kΩ) introduces measurement errors that get more and more severe as the value of the charging resistor is increased. This problem could for example be overcome using an operational amplifier with a high input impedance, but this is outside the scope of this series.

Software enhancements

It is much more interesting to try to get around these limitations in software. In particular it is possible to use a timer with a resolution of one microsecond. This increases the resolution of the capacitance measurement one thousandfold, allowing measurements in nanofarads (**Figure 7**). Library PORT.DLL provides functions TIMEINITUS and

TIMEREADUS for this purpose, where ‘US’ stands for microseconds (μs). These functions are used in **Listing 2**.

When we consider making measurements in microseconds, we must look at the effect of Windows on the timing accuracy. In principle other process running in parallel can interrupt the measurement program and cause large inaccuracies. This can be prevented by raising the priority of the measurement task, for which a special function is provided in PORT.DLL. Using REALTIME (True) we can obtain greater reliability. It is essential to set REALTIME (False) after the measurement has been taken. The exact accuracy achieved depends on the PC: with a 200 MHz Pentium MMX we measured variations of about 50 μs in the measured value; with a faster PC this may be reduced. If the same experiment is carried out in Delphi, the timing accuracy is about 20 times better. The method is described in the *Elektor Electronics* book ‘PC Interfaces under Windows’, to be published soon). It is nonetheless impressive that an interpreted language such as Visual Basic can achieve such timing accuracy.

Alongside these changes to the program, we can also improve the basic accuracy of the measurements. We have already seen two sources of error in the simple equation t/ms = C/μF, namely the threshold voltage being slightly too low and the input impedance of the DSR signal. There is a third source of error: the DTR output does not switch exactly between -10 V and +10 V, but has an internal resistance of roughly 430 Ω. The charging resistance is therefore in effect about 1430 Ω. Further, this internal resistance is non-linearly dependent on the current. The TXD output also has an internal resistance, making the voltage on the capacitor slightly higher than expected. The effects are too complicated to be analysed mathematically, and so we take the course preferred by experienced engineers in the face of a complicated calculation: test; measure; calibrate. All the errors can be condensed into a single correction factor F which can be determined with a calibration measurement. For this we require a capacitor whose value is accurately known (or which can be accurately measured). Then the correction factor can be adjusted until the reading is correct. On the author’s PC the value of F was found to be 1.19; this value can of course be used on any PC, but there will be individual variations from machine to machine which can only be compensated for by determining the correct value of F in each case.

Resistance measurement

We can measure resistance using the same method; this is similar to the way that potentiometers are read by PC games cards. The circuit for measuring resistance is shown in **Figure 8** and is essentially the same as the capacitance-measuring circuit. Here, however, we use a fixed capacitor and work with various resistor values.

To test this circuit we use the program (**Listing 3**) which measures the charging time to the highest accuracy. We use again the technique described above for measuring small capacitors: REALTIME (True) gives us good timing accuracy. **Figure 9** shows how the results appear on the screen.

The circuit can be tested using metal film resistors with a tolerance of 1 %. Measurements with an accurate ohmmeter indicate that in general the tolerance of such resistors is rather better. A set of tests using C=47 μF gave the following results:

R/kΩ	T/ms
0	33.7
0.1	34
0.22	34.5
0.56	37.9
1	45.5
2.2	76.3
4.7	147.5
6.8	204.9
8.2	245.9
10	294.6
15	433.7
22	661.2

From the numbers alone a strong linear dependency can be seen. An exact determination of the linearity is possible using a graph: an analysis using Excel produced the curve shown in **Figure 10**. It can be seen that linearity is good between around 2.2 kΩ and 10 kΩ.

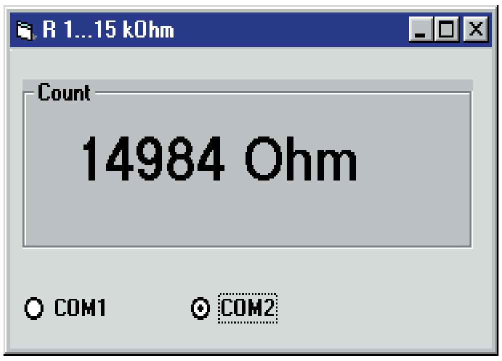


Figure 11. Measurement using a 15 kΩ resistor.

The reasons for deviation from the ideal linear curve are the same as were found in measuring capacitance. With very small charging resistances we get an error due to the non-linear internal resistance of the DTR output driver; with large resistances (say around 22 kΩ) the low impedance of the DSR input causes error.

The data obtained can be converted into an explicit equation which will let us calculate a resistance with high accuracy. **Listing 4** shows the measurement routine for resistance. The calculation in question reads:

R = 2200 + 7800 * (T - 76300) / (294600 - 76300)

Using this formula we can obtain a measurement accuracy of around 1 % in the range 1.5 kΩ to 15 kΩ. It should be borne in mind, however, that this function, specific to a particular PC, will not give the same accuracy on a different machine. The various measurements and calculations must be carried out afresh for each PC. It is sufficient to measure the charging times for two resistors, say 2.2 kΩ and 10 kΩ and substitute these in the formula. Failing this, it is possible simply to consider where the greatest sources of error lie: in this case our attention turns immediately to the electrolytic capacitor. Electrolytics often have a capacitance very different from the value printed on them, which has a significant effect on the charging time for a given resistance. If a suitable correction factor is inserted in the line

```
T = T * 1.0000000001
```

then this error will be compensated for, and reasonably usable results (see **Figure 11**) can be obtained. A correction factor of greater than or less than one will be required according to whether the capacitor has a value lower or higher than nominal.

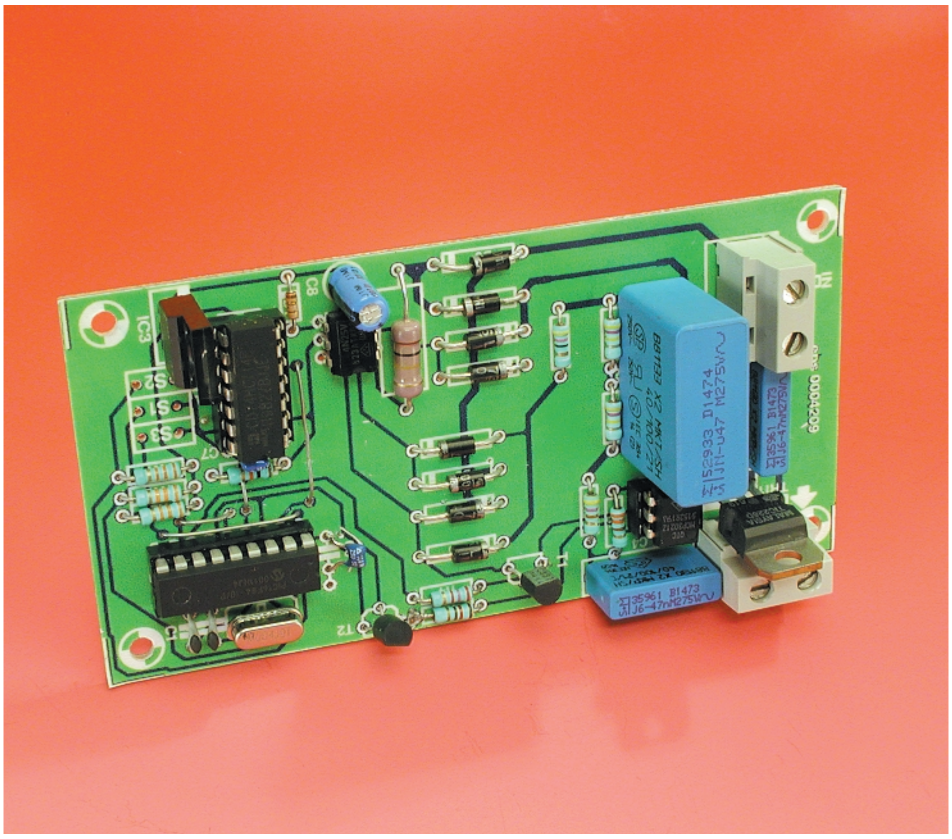
(00074-5)

Microprocessor Controlled Light Dimmer

with pushbutton control

Design by P. Staelens

This easy to construct light dimmer is controlled by a PIC controller and can handle a maximum current of 6 A. Three pushbuttons ensure extremely user-friendly operation.



The light dimmer described here can be split into two parts. The light dimmer proper follows the familiar recipe. It is an analogue circuit, which uses a triac for phase control of the mains voltage. This is hardly a surprise. A little more striking is the fact that this dimmer circuit is not controlled with a simple potentiometer but with a programmed PIC16F84 processor. The processor is operated with the aid of three push-buttons: a combined on/off button, an up and a down button.

Usage Instructions

The microprocessor is programmed in such a way that the dimmer will be in the 'off' position when first powered up, while at the same time the control circuit starts off in the

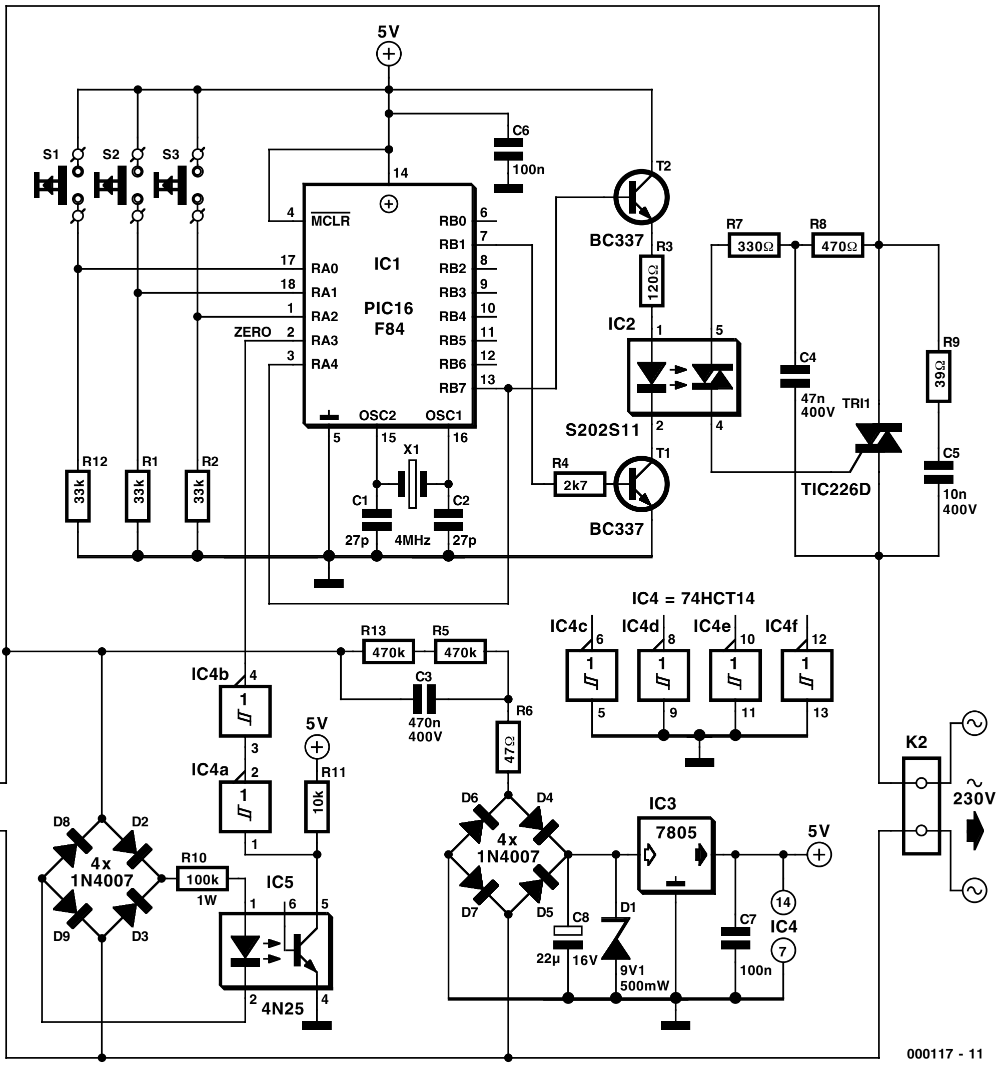


Figure 1. The classic triac light dimmer is driven from a programmed PIC16F84 processor.

'fully open' position. Pushing the on/off button activates the dimmer, only then will current be supplied to the attached lamp. However, it is also possible to adjust the control circuit before the dimmer is activated; in other words, it makes no difference whether the dimmer is activated or not.

Operation

The dimmer schematic is shown in **Figure 1**. It is immediately obvious that the amount of hardware required is quite small. The various functional blocks will be easily recognised by most of you. At top centre is the PIC-processor, to its right is the triac circuit, at bottom left is the zero-crossing detector and at bottom right is the power supply.

The **processor circuit** is a textbook example and requires no further explanation. The three input pushbuttons are connected to RA0, RA1 and RA2, while the triac circuit is driven from RB1 and RB7. The connection that has been drawn between RB7 and RA4 is now actually superfluous. Initially, the software used this connection to sense if the dimmer was on or off. Later on this was changed to enable the processor to keep track if this itself.

Now to the **triac circuit**. This is also a textbook example, with the exception that the triac (TRI1) is driven by a triac-driver (IC2). This provides separation between the low voltage and the high voltage circuitry. This is required because ground of the 5 V power supply has a different potential with respect to the high voltage circuit. T1 and T2 control the triac driver. These two transistors together perform an AND-function. The drive signal is applied only when both transistors conduct (RB1 and RB7 high). Snubber network R9/C5 protects the triac from inductive loads.

In order to determine the exact trigger point, a **zero-crossing detector** is indispensable. IC5 (optocoupler 4N25) is the most important part in this sub-circuit. A separate bridge rectifier drives the light source in the optocoupler. This is because the other bridge rectifier does not provide phase information of the mains, due to the presence of C3. The 100 Hz voltage applied to the input of IC5 causes the transistor in the optocoupler to block briefly at every zero crossing of the mains voltage. As a consequence, at every zero crossing there is a short 5-V pulse at the collector of this transistor. This signal is applied to the processor via the double Schmitt-trigger IC4a/IC4b. This makes it possible for the software to synchronise itself to the mains.

Finally, **the power supply**. As can be observed, the required 5 V supply voltage is derived directly from the mains, without the use of a transformer. 'AC resistor' C3 reduces the mains voltage to an acceptable level. This reduced voltage is rectified by a bridge rectifier (D4-D7) and regulated by a 7805. Diode D1 prevents the input voltage of IC3 from exceeding safe limits. Two 100 nF capacitors are used for power supply decoupling.

Because a 5 V power supply appears harmless, we would like to stress that this voltage in this instance is everything but safe! The reason is that this power supply is not electrically isolated from the mains. Be very careful!

Construction

Figure 2 depicts the layout for the PCB designed for the project. The construction of the light dimmer is a relatively simple task, because the number of components required is very modest, and the PCB and programmed PIC are available from the *Elektor Electronics* Readers Services (the source code files as well, for that matter).

We again urge all prospective constructors to take the utmost caution. The entire circuit is directly connected to the mains! Despite the presence of an optical triac driver and an optocoupler, the entire low voltage circuit is **not** electrically isolated from the mains. Keep this in mind while testing and making measurements on this circuit.

Another important practical matter is that for C3, C4 and C5, Class-X2 capacitors must be used. These are short-circuit proof and are guaranteed never to cause a short circuit in the event that they develop a defect. If you intend to fit the circuit into a typical enclosure, it may be better to mount voltage regulator IC3 horizontally. This can be achieved by bending IC3 over the top of IC4; do not trim the leads too short, otherwise this will not be possible.

Because of the compact dimensions of the PCB, finding a suitable enclosure should not be an issue. Use plastic screws and standoffs to fasten the PCB, in order to maintain the required insulation spacing between mains voltage and parts that may be touched. Our prototype was built into a plastic case from Bopla. The PCB fitted nicely, but it was necessary to shorten the internal stand-offs by some 3 mm for the cover to be fitted properly.

(000117-1)

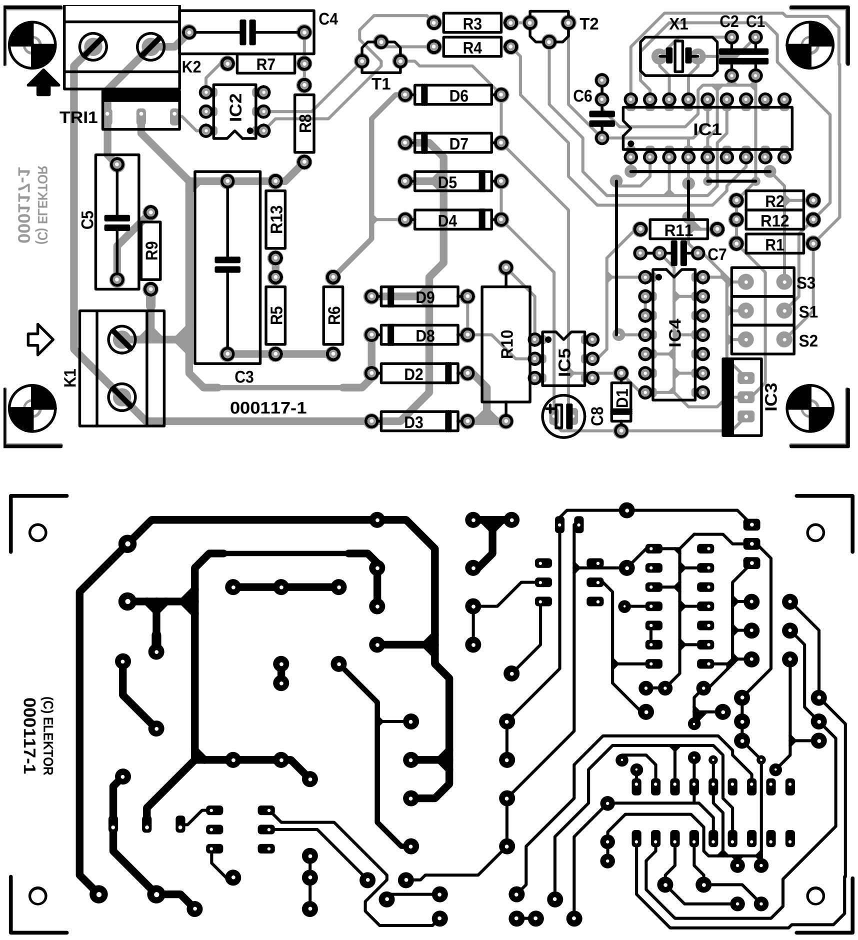


Figure 2. The PCB has everything clearly arranged and is compact. Keep in mind that the entire circuit is connected to the mains!

COMPONENTS LIST

Resistors:
R1,R2,R12 = 33 kΩ
R3 = 120 Ω
R4 = 2kΩ7
R5,R13 = 470 kΩ
R6 = 47 Ω
R7 = 330 Ω
R8 = 470 Ω
R9 = 39 Ω
R10 = 100 kΩ 1 W
R11 = 10 kΩ

Capacitors:
C1,C2 = 27 pF
C3 = 0.47µF 400 V (Class X2)
C4 = 47 nF 400 V (Class X2)
C5 = 10 nF 400 V (Class X2)
C6,C7= 100 nF
C8 = 22 µF 16 V radial

Semiconductors:
D1 = zener diode 9.1V 500mW
D2-D9 = 1N4007
T1,T2 = BC337
Tri1 = TIC226D
IC1 = PIC16F84-04, programmed, order code **000117-41** (see Readers Services page)
IC2 = MOC3021
IC3 = 7805
IC4 = 74HCT14
IC5 = 4N25

Miscellaneous:
S1,S2,S3 = pushbutton with make contact, safety Class 2
X1 =4 MHz quartz crystal
K1,K2 = 2-way PCB terminal block, lead pitch 7.5 mm
Case: e.g. Bopla (Conrad-Electronics order code 52 22 52-11)
PCB, order code **000117-1** (see Readers Services page)
Project disk (PIC source code files), order code **000117-11**

Stand-Alone MP3 Players

a new lease of life for the old PC

By H. Baggen

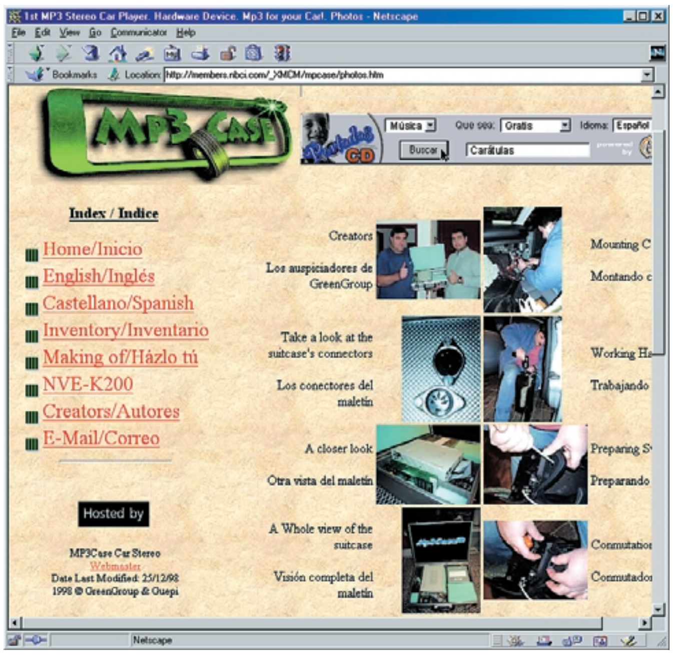
MP3 players are now in the shops in assorted colours and varieties, with different price tags, from small Walkman-like players using solid-state memories to full-blown CD players for use in the living room.

Cramming a CD-R with MP3 files is an excellent way of ensuring many hours of listening pleasure. But the real joy is, of course, found in playing home-burned CDs on a home-made MP3 CD player.

MP3 music files are compressed in such a way that they are easily transferred via a relatively slow medium like the Internet. Not surprisingly, MP3 gained its popularity entirely from the Internet. Thanks to the excellent sound quality offered by MP3 files, this particular file format has also become popular on equipment other than the PC. The arrival of portable stand-alone MP3 players gave the format a considerable boost. However, because the memory cards used in these players are relatively expensive and limited in respect of storage capacity, a new phenomenon has emerged: the MP3 CD. A home-burned CD-R of less than a pound each has enough capacity for 10 hours worth of MP3 files. Although off-the-shelf players (both portable and more residential ones) are now available for these CDs, it is, of course, more fun to build one yourself. Surprisingly, that is easily achieved with some old computer parts.

So what's needed to build my own player for MP3 CDs? The answer is: an old motherboard with at least a Pentium 100 or 133, a computer power supply, a CD-ROM drive, a sound card and (optionally) an old hard disk. On the Internet we came across several websites describing the construction of a stand-alone MP3 player from not much more than the above components.

The first example to be mentioned here is **mpMan** from Mirko Roller. Mirko built an MP3 CD player based on an old Pentium-100 motherboard. He employs a 2-line LCD as a display. At the time of writing (mid-November 2000) Mirko was busy developing a boot image for the CD, which allows a bootable MP3 CD to be burned that's capable of starting the system at power-on. A floppy disk or hard disk drive is then superfluous. The source code for this project is available from Mirko's website.


1st MP3 Stereo Car Player. Hardware Device. Mp3 for your Car!. Photos - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://members.nbci.com/_XMCM/mpcase/photos.htm
MP3 CASE
Música Que sea: Gratis Idioma: Español
Buscar Carátulas
Index / Indice
Home/Inicio
English/Inglés
Castellano/Spanish
Inventory/Inventario
Making of/Házlo tú
NVE-K200
Creators/Autores
E-Mail/Correo
Creators
Los auspiciadores de GreenGroup
Mounting C
Montando c
Take a look at the suitcase's connectors
Los conectores del maletín
Working Ha
Trabajando
A closer look
Otra vista del maletín
Preparing S
Preparando
A Whole view of the suitcase
Visión completa del maletín
Conmutatior
Conmutador
Hosted by
MP3Case Car Stereo
Webmaster
Date Last Modified: 25/12/98
1998 © GreenGroup & Guepi
Netscape

The aim of the Spanish project dubbed **MP3Case Car Stereo Player** is to bring MP3 music into your car. The project was started in 1998 and employs a souped up Pentium-133. The system is built in a separate case, and WinAmp is used as the MP3 program. Even a remote control is catered for. Although fairly extensive construction notes and a parts list are available, not everything is to be had in English. If you are really interested in this design, we reckon there's no way to avoid brushing up your Spanish or seek the help of a translator.

Though **Andie's MP3 CD Player Project** is also based on 'scrap' PC components, he manages to cram it all into a case of the size of regular CD player with a height of just 7 cm. The case then contains a CD-ROM drive as well as a 20-GB hard disk drive which easily holds up to 340 hours worth of music. A nice detail of this design is that it incorporates a webserver, allowing the user to compile and edit playlists using a web browser.

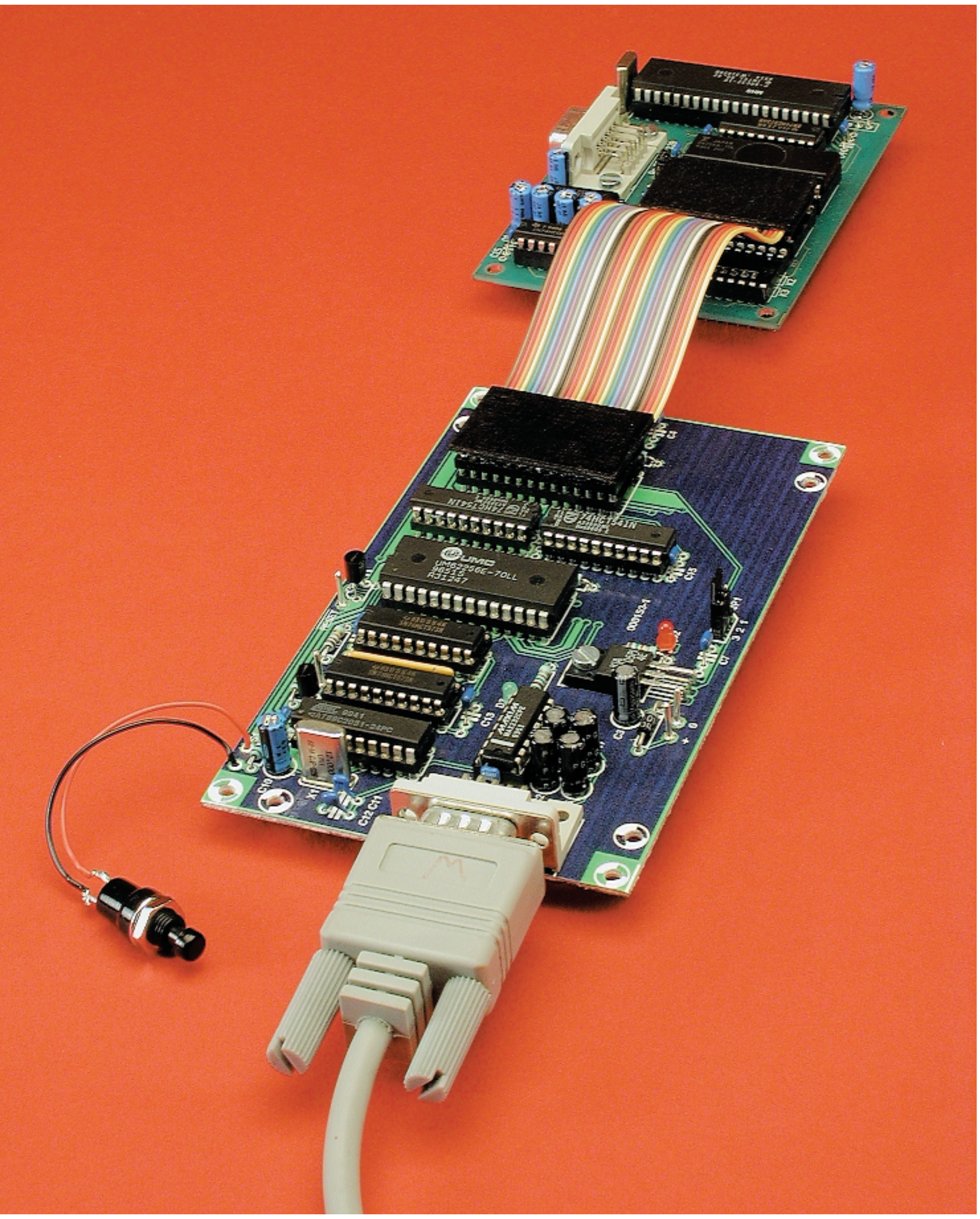
(015005-1)

mpMan - Build your own MP3 player:
http://www.dvz.fh-koeln.de/~bn520/mp3.html
MP3Case Car Stereo Player:
http://members.nbci.com/_XMCM/mpcase/english.htm
Andie's MP3-CD-Player Project:
http://homepages.compuserve.de/asdevel/mp3-cd-player.htm

Emulator for 27C256 EPROM

with RS232 control

Design by B. Legrand and D. Mautaulon



Going through our archives we discovered that it's been almost ten years since we published an EPROM emulator. The version we propose in this article should meet today's demands of hobbyists wishing to debug microcontroller systems based on an EPROM. We have chosen the 27C256 because it is cheap and currently the most widely used EPROM in the hobby area.

Technical features

- Emulates the most widely used EPROM type 27C256
- RS232 controlled
- Employs HyperTerminal for data transfer between PC and EPROM simulator.
- Recognizes industry-standard IntelHex format
- May be adapted to suit 27C64 and 27C128 by modifying circuit around pins 26 and 27. Support for 27C512 also possible with some hardware and software modifications.

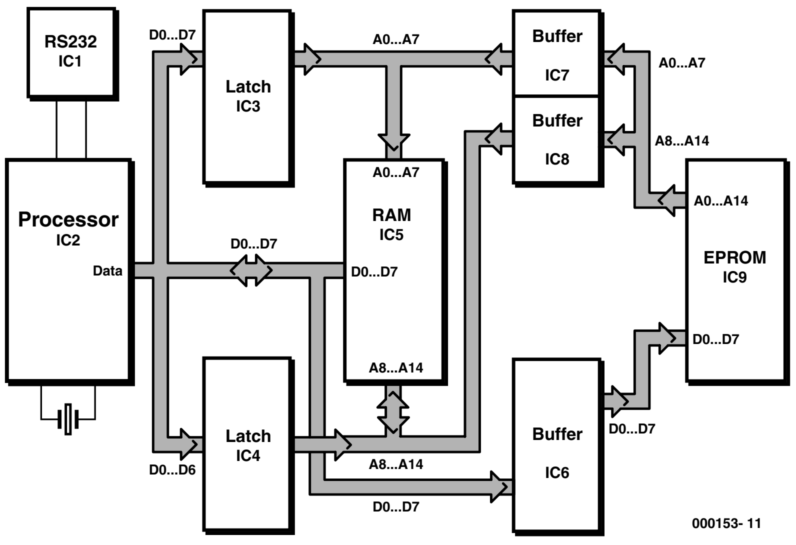


Figure 1. Block diagram of the 27C256 EPROM emulator, with the Atmel microcontroller clearly at the hub of things.

An EPROM emulator is a development tool designed to facilitate code debugging and code writing jobs on circuits incorporating an EPROM (electrically erasable read only memory). An EPROM, as most of you will know, can not be reprogrammed before its previous contents has been erased through exposure to a certain amount of ultraviolet (UV) light. So, even for the smallest modification to the code in your EPROM, you need to do a complete erase-and-reprogram cycle, which is tedious and costly given the time lost and the price of an UV eraser box.

An EPROM emulator obviates these problems by allowing you to debug, rewrite and download code as many times as you like, until the desired system operation is achieved, all without having to erase a single EPROM. Having extensively tested the target program, you need to program an EPROM just once yet rest assured that it will work as planned.

Meanwhile, in this day and age of Flash reprogrammable and ISP (in-system programmable) devices, it is fair to reflect on the advantages, if any, of an EPROM emulator. Also, one can not fail to recognise the trend towards ever larger memory capacities.

Despite the above trends, there is still a fair number of circuits based on microcontrollers running code from an external EPROM. These controllers include devices from the 8051 series, the 68HC11 and 80C5xx. It is precisely in this area where the present emulator will be highly valued.

Principle of operation

The block diagram shown in **Figure 1** is classic and typical for this type of application.

The underlying principle of an EPROM emulator is that it replaces ‘dead’ memory like ROM or EPROM by ‘live’ memory (RAM) with double (two-port) access. The RAM is flanked by two latches and supported by a microcontroller.

As in any EPROM emulator that’s any good, the heart of the circuit is a RAM device (here, IC5) which essentially replaces the (E)PROM of the target (or ‘host’) system. The RAM is surrounded by latches (IC3 and IC4) and buffers (IC6, IC7 and IC8). The exact function of these components will be discussed a bit further on.

The distinctive feature of the present EPROM emulator (as compared with traditional designs) is the presence of a microcontroller (IC2). This component looks after the control of the latches and the RAM, and also handles the correct reception (from the PC) and processing of the object code to be transferred to the target system. Here, an Atmel 89AT2051 microcontroller is used. The main reason for choosing this device is that it offers on-chip serial I/O as well as EPROM to store the firmware that handles the code reception and control of the emulator.

Practical circuit

By and large, the circuit diagram shown in **Figure 2** follows the general structure of the block diagram. In fact, all building blocks discussed above are fairly easy to find back in the schematic.

IC1, a MAX232, handles the classic function of converting the ±10-V signal levels on the RS232 lines to logic levels (+5 V/0 V), in two directions. The MAX232 allows the microcontroller in the circuit to communicate with the PC via the RS232 port. Although many PCs these days are capable of handling serial signals with a swing of 5 V, simply omitting the MAX232 would require the RxD and TxD signals to be inverted!

IC2 is the microcontroller type AT89C2051. It controls latches IC3 and IC4 as well as buffers IC6, IC7 and IC8. These integrated circuits ensure the correct interfacing with RAM IC5. In this context, ‘correct’ means that the RAM may be accessed by either the microcontroller or the external (i.e., target) circuit, but never at the same time.

The RAM in fact emulates (mimics) the (E)PROM which has been removed form the target circuit. When the target circuit has access to the RAM, it will behave as if a system (E)PROM was installed, hence the term ‘emulator’. To be able to pull off this trick, the RAM requires two peripheral devices. From

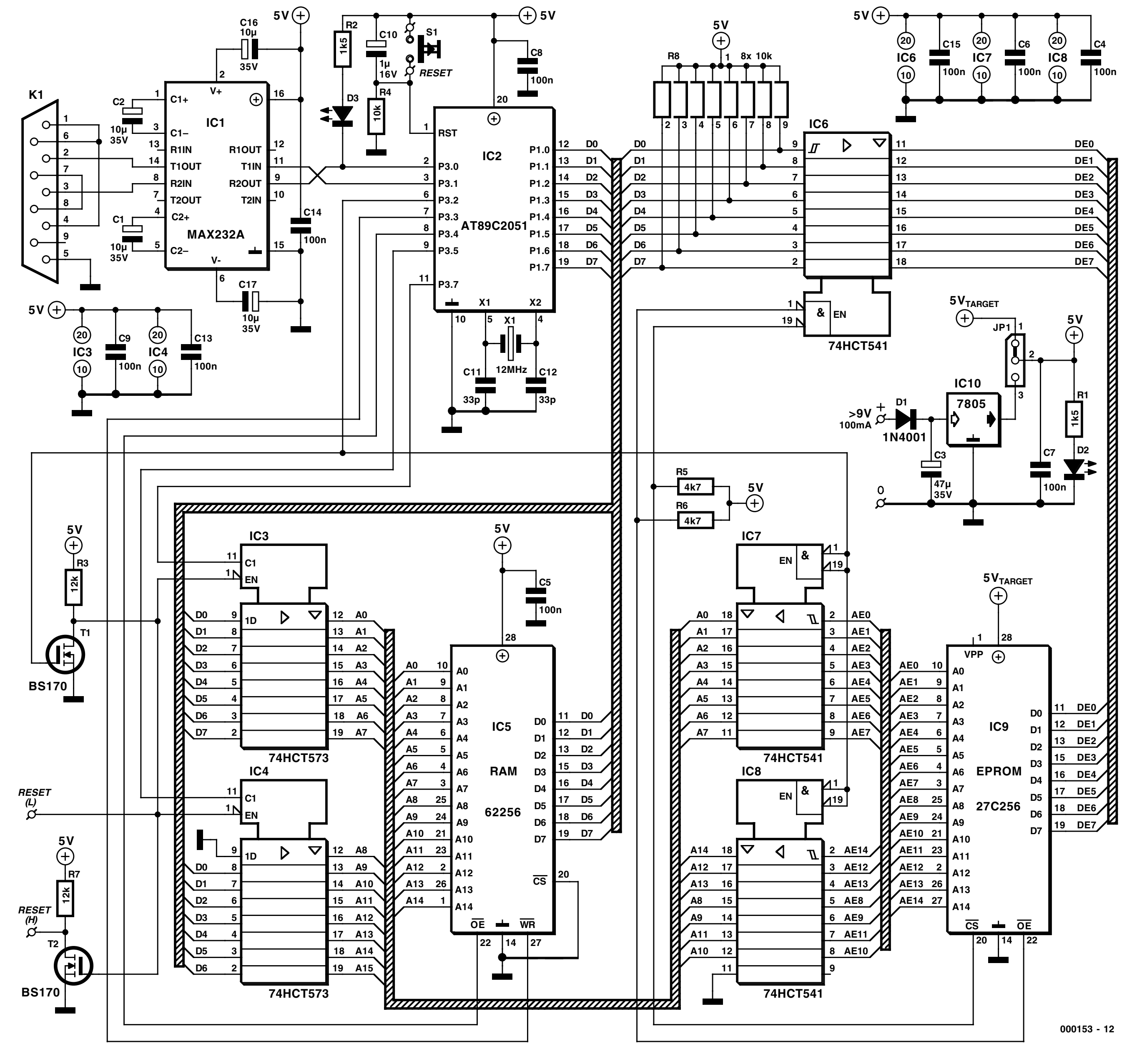


Figure 2. Circuit diagram of the 27C256 EPROM emulator.

one side, the emulator electronics enables the object code under test to be written into the RAM, while from the other side the external (host) system can access the RAM to read code which the host microcontroller will eventually execute.

Latches IC3 and IC4 connect the host system address lines to the RAM. Depending on commands received from the microcontroller, these ICs transfer the data on the internal databus, D0-D7. Each of the latch enable inputs is controlled by an individual I/O line of the central processor. This configuration allows the processor to control the RAM address bus. Once the target program is stored in RAM, the two latches go into high-impedance mode to avoid contention problems when the system switches to emulation mode, i.e., with the RAM effectively connected into the target system.

Buffers IC6, IC7 and IC8 ensure the quasi-connection of the RAM address and datalines to the external (host) circuit.

Power supply

The 27C256 EPROM emulator may be powered in two ways. The first, which we will treat as the 'standard' method, consists of powering the circuit by means of a mains adapter.

The on-board power supply consists of a three-pin voltage regulator type 7805 (IC10) in a classic configuration. This component provides the +5-V regulated supply voltage to the emulator circuitry. Because only 100 mA or so of output current is required, the 7805 has an undemanding job in this circuit. Diode D1 protects the circuit against reverse polarised input voltages. LED D2 acts as a power on/off indicator.

The second method consists of powering the emulator from the target (host) system, which will be possible in most cases because +5 volts will be present for the digital circuitry around the (E)PROM. If you envisage using this method all the time, you may omit components IC10, diode D1 and capacitor C3 from the emulator circuit.

Jumper JP1 (located near the voltage regulator) allows you to select between internal and external powering.

The serial link

The communication between the PC and the emulator consists of two-way traffic via the RS232 port, for which suitable circuitry and cable lines have to be present.

On the emulator board, the RS232 interface consists of a MAX232 (IC1) in its standard application circuit with four pump capacitors. Sure, we could have used the SMA version of the MAX232 and enclosed the complete serial interface in a sub-D connector case for easy connection the PC. However, to keep construction as easy as possible we decided to fit all the parts that make up the interface on the emulator board. This choice also enables an off the shelf serial cable to be used.

A note about the RS232 link — this should consist of a **standard** RS232 cable, i.e., not one with 'crossed wires' (also known as null/zero-modem cable).

How it works

The operation of the EPROM emulator may be divided into two phases: (1) loading the RAM, and (2) simulating an (E)PROM in the target system.

For the first task, the PC transmits, via its serial port, the hexadecimal code to be stored in the pseudo EPROM. For the second function, if the target system is to gain access to the code, it is necessary for the microcontroller to pull its port line P3.2 logic Low to actuate 3-state drivers IC7 and IC8. Two FETs, T1 and T2, keep the host system in the reset state.

Let's see how this works in more detail. At power-on, the microcontroller, IC2, prepares all circuitry for data to be written into the RAM. This is done by pulling all lines of port P3 to logic High, with the exception of P3.5 and P3.7.

For the microcontroller to get control over the RAM it has to pull port line P3.2 to logic High.

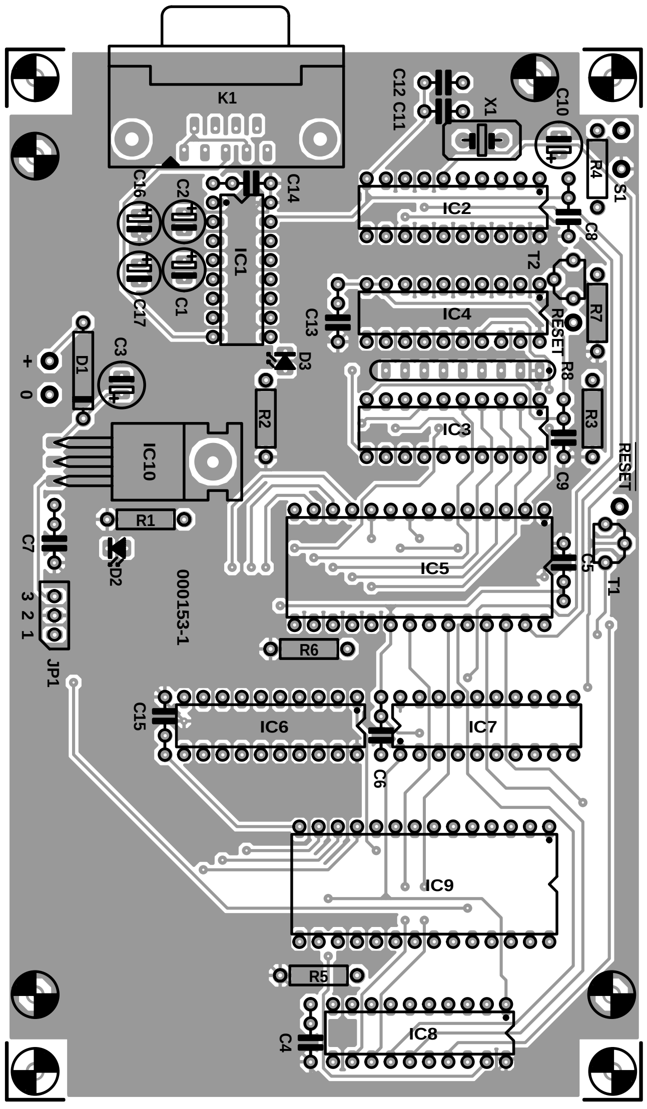


Figure 3a. Component mounting plan of the circuit board designed for the 27C256 EPROM emulator.

COMPONENTS LIST

Resistors:
R1,R2 = 1kΩ5
R3,R7 = 12kΩ
R4 = 10kΩ
R5,R6 = 4kΩ7
R8 = 10kΩ 8-way SIL array

Capacitors:
C1,C2,C16,C17 = 10µF 35V
C3 =47µF 35V
C4-C9,C13-C15 =100nF
C10 =1µF 16V
C11,C12 =33pF

Semiconductors:
D1 = 1N4001
D2,D3 = high efficiency LED
T1,T2 = BS170
IC1 = MAX232 (Maxim)
IC2 = AT89C2051 (Atmel), programmed, order code **000153-41**
IC3,IC4 = 74HCT573
IC5 = 62256 (RAM)
IC6,IC7,IC8 = 74HCT541
IC9 = EPROM being emulated
IC10 = 7805

Miscellaneous:
K1=9-way sub-D socket (female), PCB mount
PC1-PC4= solder pin
JP1= 3-way SIL pinheader with jumper
S1= pushbutton, 1 make contact
X1=12MHz quartz crystal
PCB, order code 000153-1 (see Readers Services page)
Disk, project software, order code **000153-11** (see Readers Services page)

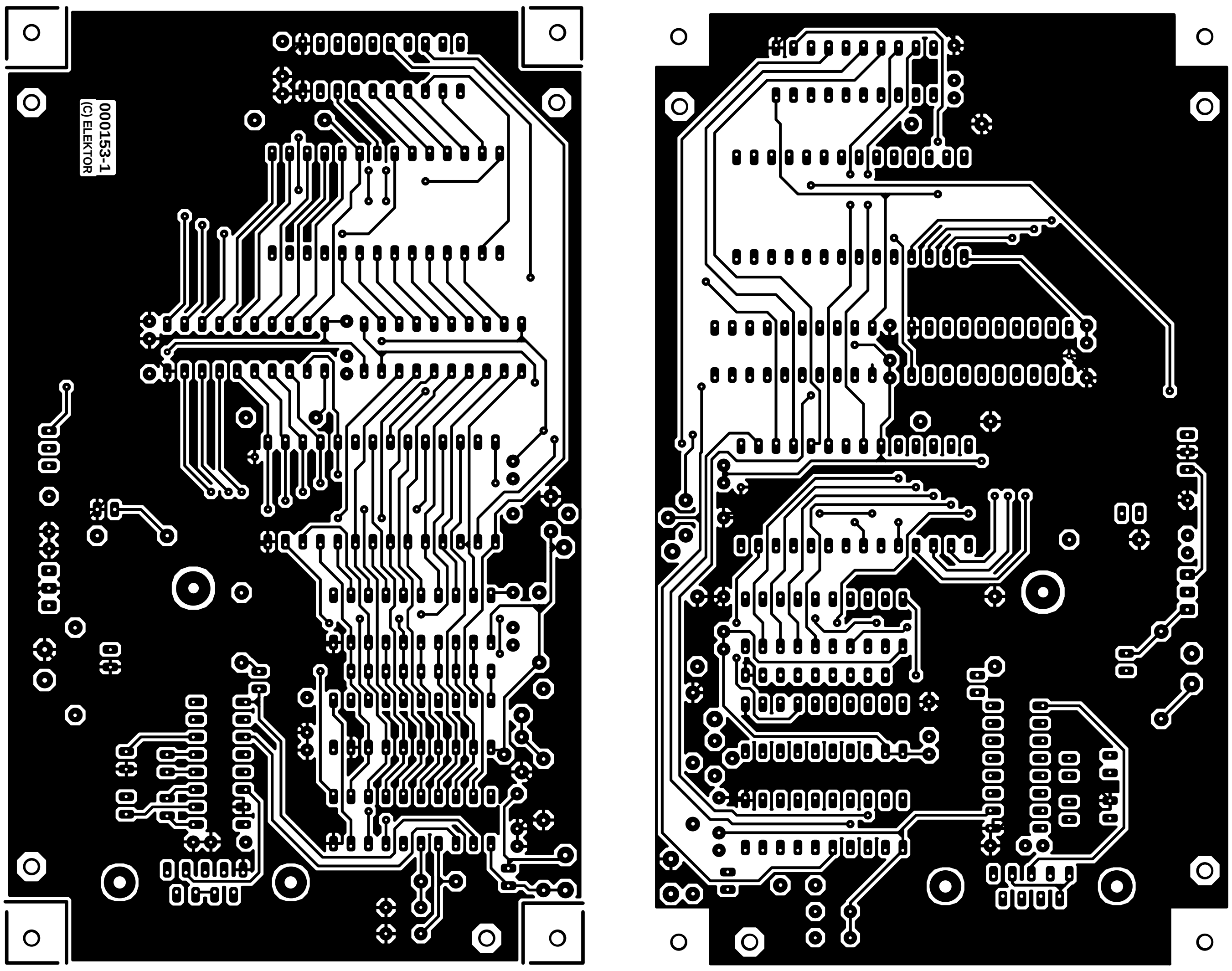


Figure 3b. Copper track layout of the circuit board designed for the 27C256 EPROM emulator. This board is double-sided and through-plated.

To be able to load the RAM, port P1 is supplied with the high address of the first databyte. Next, the 3-state driver IC4 is opened and closed again by means of port line P3.5 in order to block this address.

The above sequence is repeated for the low address, this time with the aid of port line P3.7 controlling another 3-state driver, IC3.

Port line P3.2 of the AT89S2015 microcontroller is programmed to switch the outputs of buffers IC6, IC7 and IC8 to high-impedance (tri-state), which is necessary to ward off all disturbances caused by the external electronics from the RAM during the write process.

The same signal is also inverted by the combination T1-R3. The inverted control signal serves to actuate IC3 and IC4 in such a way that the RAM address lines are properly driven. The control signal on P3.2 is put to the disposal of the target circuit by a pair of solder pins, RESET(L) and RESET(H). One of these signals may be used to keep the target system in the reset state while the RAM is being filed with object code.

Once the complete object code file has landed in the RAM, the microcontroller in the emulator produces a message on the RS232 port.

The PC has to send the object code file in IntelHex format, via its RS232 port. The processor on the emulator board looks after the correct reception of the file (LED D3 will light while data is being received from the PC), and arranges for each databyte to be written into the RAM at the proper location. This is achieved by IC2 copying address lines A0-A7 on to port P1 and when done producing a pulse on P3.7. Latch IC3 copies this word. The same process is repeated with address group A8-A13. The data transferred by these addresses are latched in IC4 when a pulse appears on port line P3.5. Finally, the actual databyte is copied on to P1, followed by a Low pulse produced on port line P3.3. The latter drives the write ($\overline{WR}$) input of the RAM. When a falling pulse edge appears at this input, the RAM transfers the dataword on port P1 to the specified address.

The same procedure is followed for the transfer of all datawords that make up the object code.

Once the complete IntelHex file has been received, the central processor switches the circuit to simulation mode. More specifically, the RAM is switched to read mode by pulling port line P3.3 Low and enabling the RAM output drivers by pulling $\overline{OE}$ (output enable) Low. This is achieved by controlling P3.4.

Pulling P3.3 logic Low also

enables the outputs of buffers IC7 and IC8, plus it switches the latch outputs to high-Z (tri-state) by means of the $\overline{\text{EN}}$ (enable) inputs.

In addition to these important functions, the P3.3 signal disables the two RESET outputs of the circuit. With one the two RESET outputs suitably connected to the target circuit, this will be automatically held in its reset state while the RAM is being loaded with the object code file. Once the RAM is filled, the target circuit is automatically re-initialised (very handy if the target circuit does not have a dedicated reset button or similar).

Port line P3.4 drives the RAM in such a way that the memory chip constantly places data on the internal databus.

To prevent the RAM from supplying data on the external databus, the two enable inputs of buffer IC6 are connected to $\overline{\text{OE}}$ (output enable) and $\overline{\text{CS}}$ (chip select) lines of the external electronics. This approach guarantees the correct transfer of data from the RAM to the external databus whenever the target system addresses the EPROM simulator.

If new data has to be written into the RAM, you have to press the RESET button to start the file loading process.

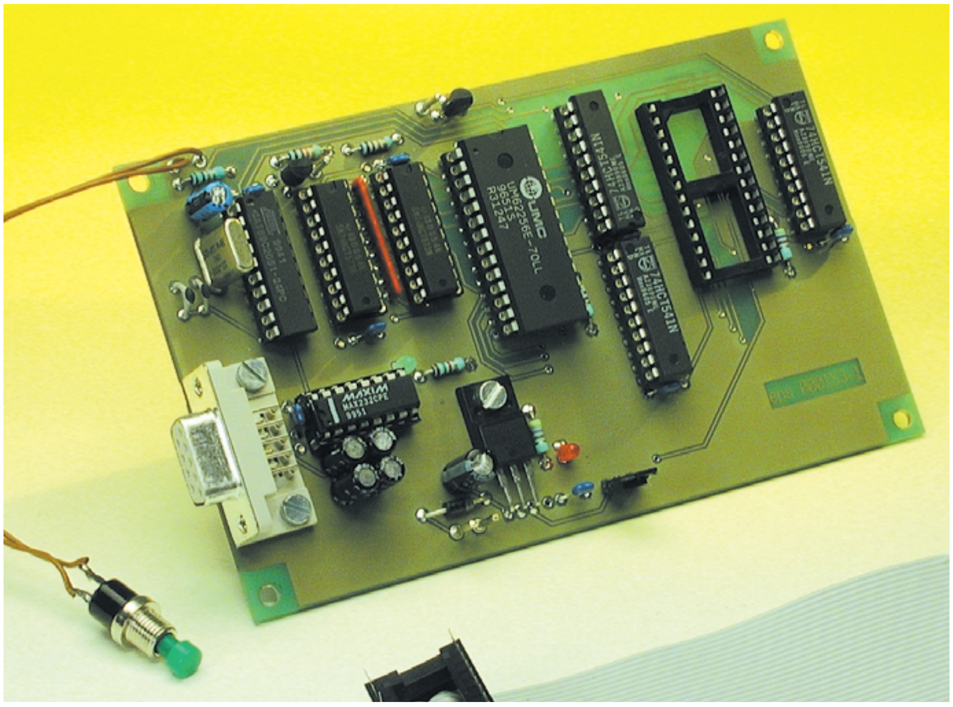


Figure 4. Finished prototype of 27C256 EPROM emulator.

Building the EPROM Emulator

As evidenced by the introductory photograph with this article, it would have been possible to make the emulator board even more compact, for example, by 'moving' the MAX232 serial interface to the serial connector casing. As discussed above, this option was not followed to ensure that everyone can build the present circuit from regular size components.

The copper track layout and component mounting plan of the emulator board are given in **Figure 3**. Fitting the parts on the board should not cause problems and a spot-on working construction, we feel, should be within the capacity of most of our readers. The board being double-sided and through-plated, it does not have a single wire link, which otherwise is the most frequently forgotten 'component'!

As customary with this type of project, it is best to start with the low-profile components like resistors, capacitors and transistors. Pay attention to the orientation of SIL resistor array R8, of which the pin with the dot (indicating the common connection) should be at the edge of the board. To cut costs, you may fit only three (high quality) IC sockets, one for the processor, one for the RAM and one for flatcable between the emulator and the EPROM in the target circuit. Note that IC7 is fitted the other way around as compared with the other ICs on the board (except IC1 and IC10).

Give the circuit a thorough check, including a supply voltage check on all ICs, before fitting the processor and the RAM into their sockets. LED D2 will light to indicate the presence of the supply voltage.

The software

The program stored in the Atmel microcontroller has been written to allow the Windows HyperTerminal communications utility to talk to the EPROM emulator.

The code transfer from the PC to the emulator is via a serial connection running at 4,800 bit/s.

The emulator recognises Intel Hexadecimal (a.k.a. IntelHex) which is a widely used format for object code file transfer between PCs and programmers. HyperTerminal is part and parcel of Windows 95 and 98 so everyone running Windows on his/her PC should have it (you'll find it under Programs → Accessories → Communications). The IntelHex file is transferred using the ASCII transmission mode (and not, as you may have expected, a protocol like Kermit or Z-modem).

The communication parameters are set to 4800 bits/s, 8 Databits, No parity, 1 Stop bit (4800,N,8,1)

In HyperTerminal, select the function 'Send Textfile'.

It is also possible to use DOS for the communication between the PC and the emulator. **DOS users** may use this command line:

```
COPY INTEL.HEX COM1:
```

Linux users will typically employ

```
Cat INTEL.HEX \dev\xxx
```

Where xxx is the port to which the EPROM emulator is connected.

(000153-1)

For further reading:

EPROM Emulator II,
B.C. Zschocke and N. Breidohr,
Elektor Electronics
July/August and September 1992.

GBPB — GameBoy Prototyping Board

A development aid for the GameBoy

Design by L. Lemmens

In previous editions we proved beyond doubt that the Nintendo GameBoy is perfectly suitable for more serious tasks than playing games. The sampling oscilloscope described in the October and November 2000 issues was a prime example. In the December 2000 issue, we published a follow-up article on GameBoy Development Tools. This month it's time to offer help in the hardware department.

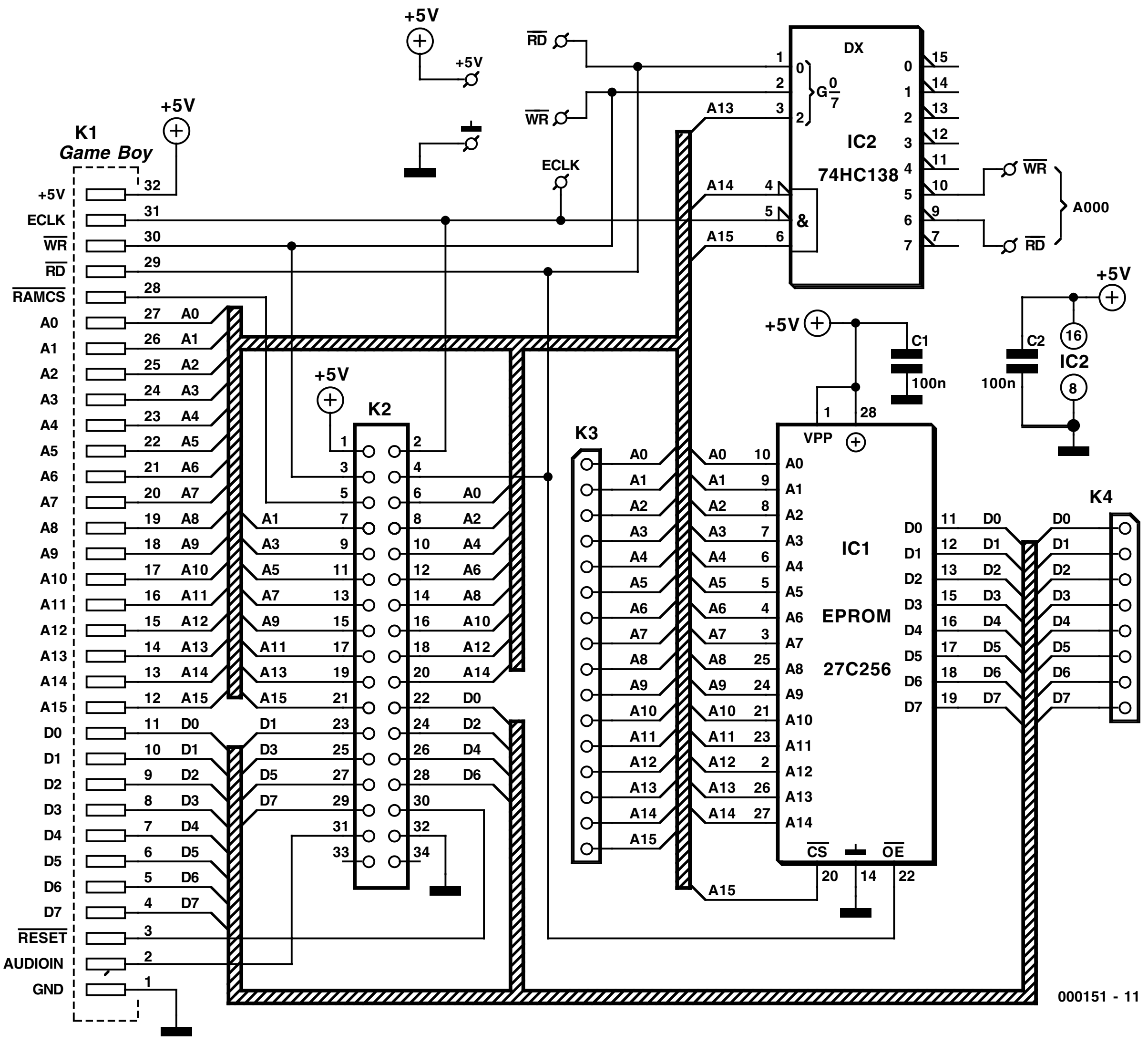


Figure 1. Circuit diagram of the GameBoy Prototyping Board (GBPB)

In the December 2000 issue of *Elektor Electronics* we supplied pointers to a variety of GameBoy emulators that allow software for this extremely popular 'games console' to be tested on a PC. Emulators are fine tools for projects consisting of software only. However, the moment extra hardware has to be linked to the GameBoy, two problems occur: the connector for the (game) cartridge is a Nintendo proprietary design and to make matters worse it is hidden deep in the GameBoy case.

These problems with the card edge connector do not appear when a PCB is designed and etched straight away for the prototype of the circuit you have in mind. Still, because a large part of the cartridge board is hidden by the GameBoy case, it will be hard, if not impossible, to do any measurements on the new board. Sure, it is possible to open the GameBoy case by removing the screws that hold the rear panel. That way you have access to the cartridge board, at least for measurements. However, opening the GameBoy case is a far cry from the ideal solution because the screws are difficult to remove with an ordinary screwdriver. Moreover, opening the GameBoy case voids the product warranty you may have, and without the rear panel there is no battery case! The most annoying problem is, however, that the prototype is at the rear side of the GameBoy, while the display and controls are located on the front (now who did that design?).

A helper board

All of the problems mentioned above are avoided by our prototyping board of which the circuit diagram is shown in **Figure 1**, and the PCB design, in **Figure 2**.

Any components on the prototyping board will be visible and accessible when the GameBoy lies horizontally on your desk. This allows measurements to be carried out on the prototype hardware while the GameBoy display and buttons are accessible at the same time.

The cartridge connector is part of the PCB, and its connections are linked to a standard 34-way boxheader. In many cases, this is sufficient, because it is then easy to connect your own (experimental) board to the GameBoy via a length of standard flatcable. The order of the connections is given on **Table 1**. Note that the boxheader has 34 pins while

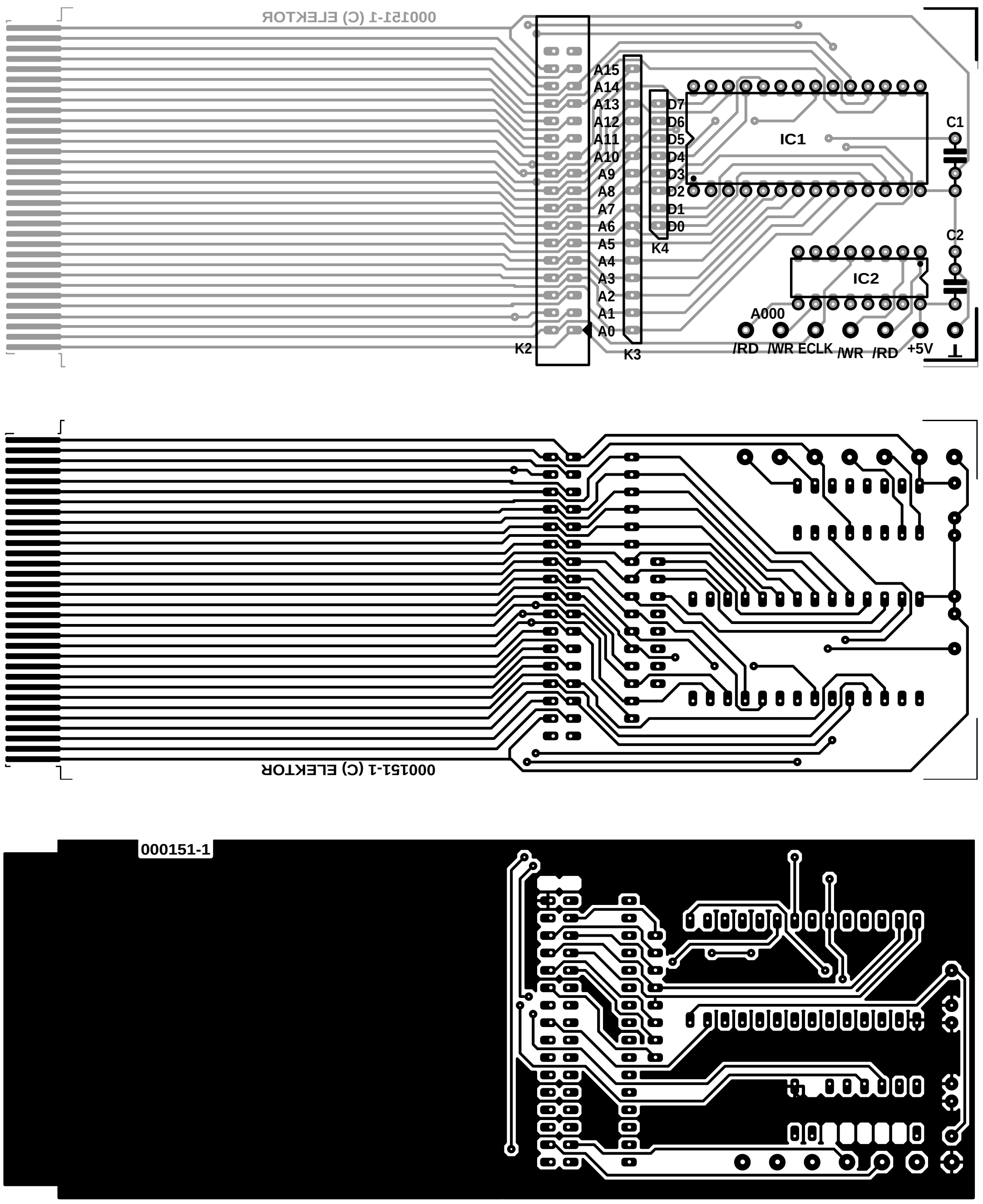


Figure 2. PCB design.

COMPONENTS LIST

Capacitors:
C1,C2 = 100nF

Semiconductors:
IC1 = socket for 27C256
IC2 = 74HC138

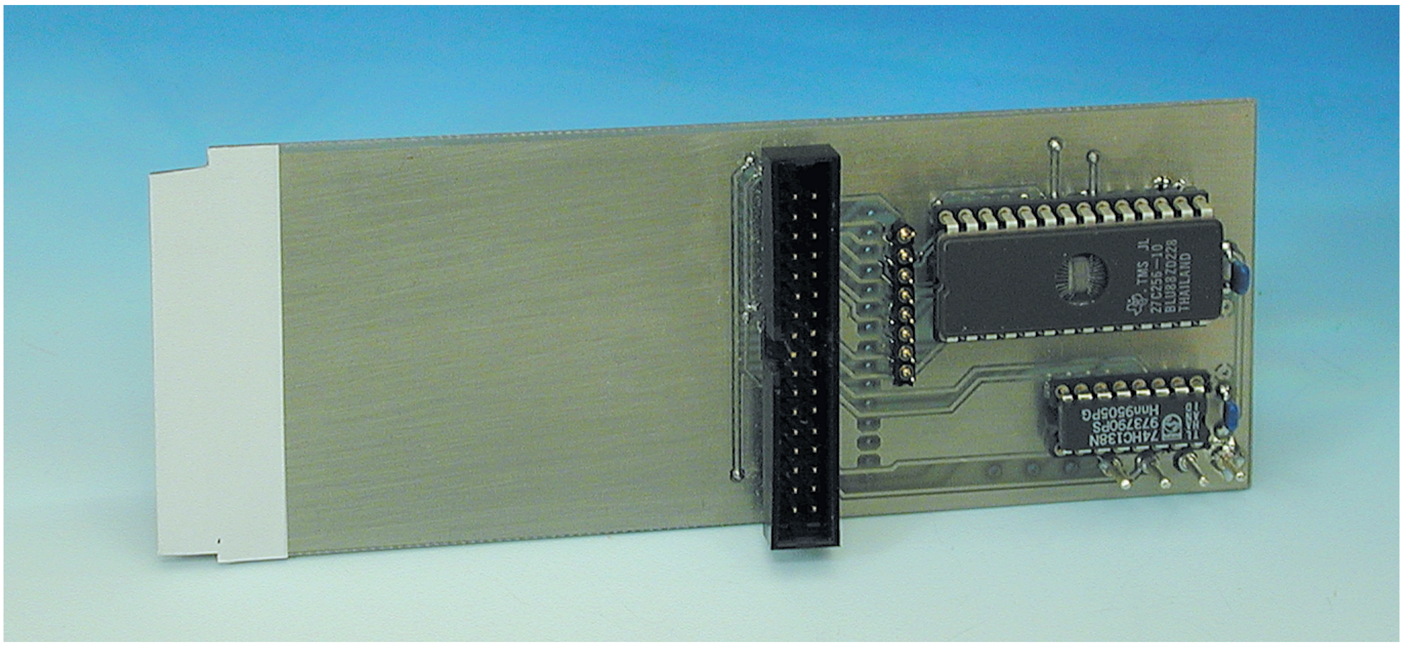
Miscellaneous:
(K1 = card edge connector on board)
K2 = 34-way boxheader
K3 = 16-way SIL-header
K4 = 8-way SIL-header
PCB, order code **000151-1** (see Readers Services page)

the GameBoy cartridge connector has only 32. The simple reason for choosing a 34-way boxheader is that a 32-way version does not exist.

Extension cards like the one described here usually have a prototyping area where additional components may be fitted to test a particular circuit you have in mind. Such a prototyping area has been purposely omitted here. The essential point in this case is to make the special cartridge connector and the extension part accessible well outside the GameBoy case. Veroboard is cheap and widely available from local electronics shops (or by mail order), and easy to connect via a piece of flatcable. What's more, despite accurate soldering and desoldering, prototyping board will not last long when used frequently.

Options

Connectors K3 and K4 are initially intended to facilitate measurements on the data and address bus. You'll



find the data and address lines neatly arranged on these contact strips. Of course, the lines may be used for external connections. Similar considerations apply to the PCB pins for $\overline{RD}$, $\overline{WR}$, ECLK and the two supply connections +5 V and GND.

The latter pins should only be used for supply voltage measurements, or for powering an external circuit. They must **never** be used to power the GameBoy or the hardware on the extension card.

Any cartridge should contain a piece of software in ROM that allows the GameBoy to start up. That's why a socket for the 27C256 EPROM is provided on the extension board. The memory may of course also be located on the board connected to K2. In that case, IC1 should not be fitted on the extension board.

An EPROM emulator like the one described elsewhere in this issue is conveniently coupled to the socket for IC1. Such a configuration is very useful and handy when you are into developing software for the GameBoy — write and assemble the source code on the PC, load the object code in the emulator using a HEX file transmitted via the serial port, and finally test the program on the GameBoy. Arguably, this is a much faster process than erasing and reprogramming an EPROM after every modification made to the program.

The circuit includes a 74HC138 (IC2) which allows two selection lines to be decoded for controlling external hardware. Pins 9 and 10 of the '138 carry the active-Low read and write signals respectively for address A000. For ease of connection, these signals are brought out to PCB pins.

Construction

Before the soldering iron is powered up, you should check if the board fits in the GameBoy. Note that the normal thickness of the cartridge case plus board should be about 2.5 mm to ensure proper contact with the GameBoy cartridge connector. Here, we only employ the PCB (i.e., no cartridge case), and that is why the PCBs supplied ready-made through our Readers Services have been produced with a special thickness. If you want to make your own board from the layout shown in Figure 2, but do not have PCB material of the right thickness, a thinner size may be employed. Strips of cardboard or plastic may be glued on to the bottom side of the board to achieve the same effect. This should be done with care and precision — if the connector is too thin, the contact with the GameBoy will be unreliable ore even totally absent. A too thick connector, on the other hand, means a serious risk of damaging the GameBoy connector beyond repair.

Populating the board will not cause problems. You should, however, consider how the board will be used, that is, with the EPROM mounted on it or with the memory device located on the external board. In many cases, boxheader K2 will suffice. If you are only into developing and testing software for the GameBoy, it is sufficient to fit the EPROM socket only.

(000151-1)

Table I. Connections on the extension board.

K2 pin #	signal
1	VCC
2	ECLK
3	$\overline{WR}$
4	$\overline{RD}$
5	$\overline{RAMCS}$
6 - 21	A0 - A15
22 - 29	D0 - D7
30	$\overline{RESET}$
31	Audio input
32	GND
33, 34	not connected

K3 pin #	signal
1 - 15	A0 - A15

K4 pin #	signal
1 - 8	D0 - D7

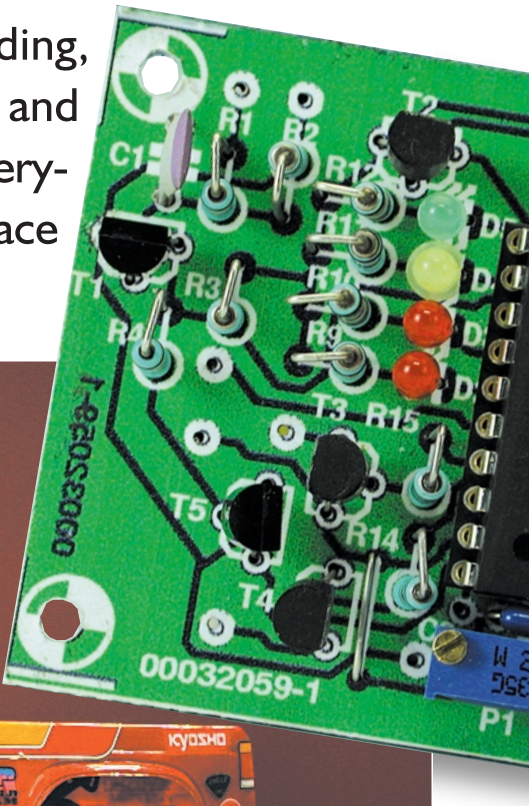
Modular speed controller for R/C models

Speed Controller Duet

part 1: overview and SpeedControl

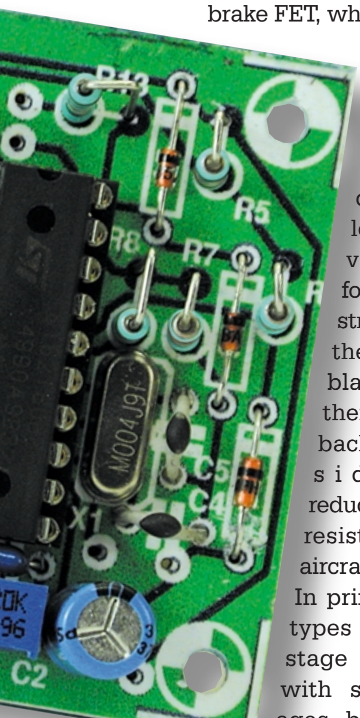
Design by B. Stuurman

The interest in speed controllers for R/C models is still just as great as ever, and we can confidently state that the Speed Controller Duet sets a new standard in this area. It is a universal system that can be used in all branches of R/C model building, including electrically powered aeroplanes, boats and cars. An equally important feature is that almost everything is programmable, thanks to an RS232 interface that allows the controller to be linked to a PC.



The Speed Controller Duet is a modular system. Its intelligent heart is formed by the SpeedControl unit. This microcontroller-based circuit does not contain any power output stages. These are instead built as separate units that are controlled by the SpeedControl. This division of functions is actually quite logical. After all, processing the received pulses has little in common with driving electric motors with currents of several tens of ampères.

There are two types of power stages for the Speed Controller Duet. SpeedPower 1 is equipped with two relays for reversing the polarity of the output voltage, for driving or sailing in reverse. This unit is thus a perfect fit for applications in model ships and cars. SpeedPower 2 is fitted with a 'brake FET'. This can be used to collapse the folding propeller of an electric glider. The principle is simple: the motor is short-circuited by the brake FET, which causes it to quickly slow down. Once the rpm's have dropped to a low enough value, the force of the air stream against the propeller blades causes them to fold back. This considerably reduces the wind resistance of the aircraft.



In principle, both types of power stage are usable with supply voltages between 7.2 and 12 V, with a maximum current of 50 A. However, the relays in the SpeedPower 1 are rated at 25 A, so it is a good idea not to go too far beyond this limit. The SpeedPower 1 and SpeedPower 2 are both fitted with a Battery Eliminator Circuit (BEC) rated at 5 V and 1.5 A, which allows the supply battery for the receiver to be eliminated.

A particularly interesting aspect of this modular approach is that several power units can be connected to the SpeedControl. For instance, each motor of a multi-motor aeroplane could be fitted with its own power unit in the motor nacelle. The available power can thus be doubled in this manner.

The power units are fitted with fast digital optocouplers, which allows the inputs and outputs of all power units of the same type to be connected in parallel, and the distance to the SpeedControl can in theory amount to several metres.

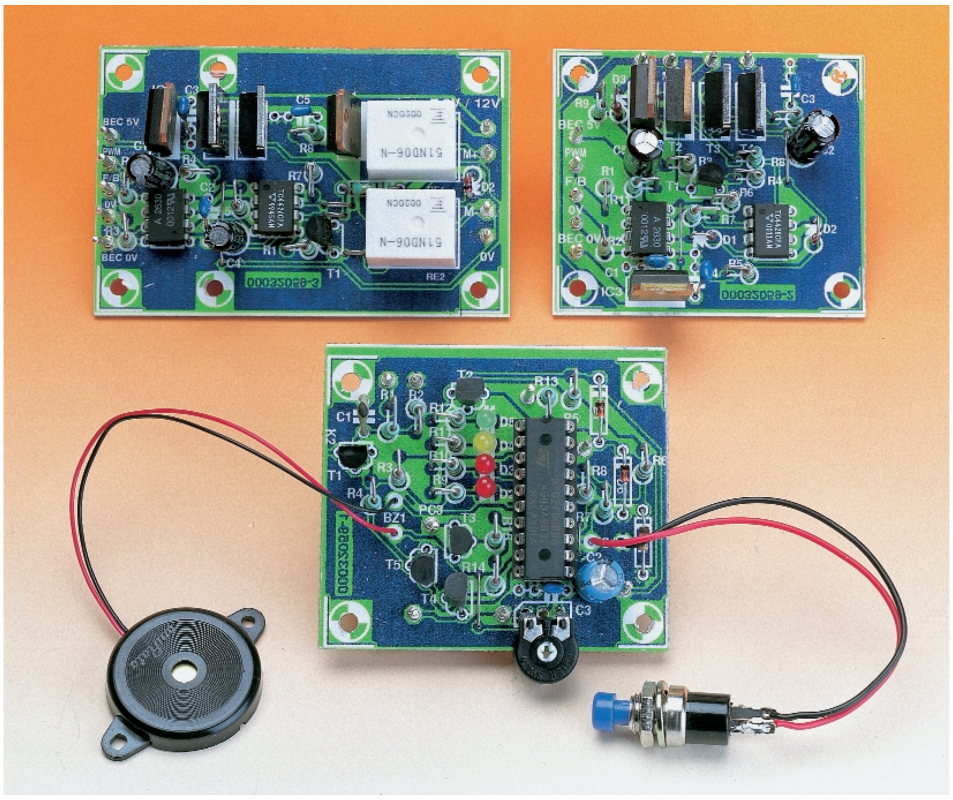


Figure 1. A bird's eye view of the Speed Control Duet. The SpeedControl is in the foreground, with the SpeedPower 1 at the left rear and the SpeedPower 2 at the right rear.

SpeedControl specifications

Maximum voltage:	6.5 V
Minimum voltage:	3.75 V
Current consumption:	10 mA (1 LED on, U=4.5 V)
Input:	
polarity:	positive
minimum pulse duration:	0.5 ms
maximum pulse duration:	2.7 ms
minimum repetition interval:	16.6 ms (60 Hz)
maximum repetition interval:	28.6 ms (35 Hz)
Outputs:	
1: PWM:	positive duty cycle 0-100%
2: for/back:	1 = reverse
Transmitter adaptation:	intelligent three-point timing
User programming:	using a PC, via RS232
Protocol:	19200, 7, n, 2
Options:	practically everything is programmable
BEC:	yes, via power unit; 5 V / 1.5 A
Cut-off:	yes, adjustable
Digital watchdog:	yes
Gliding motor control:	yes, approx. 0.7 s for 100%
Stand-alone option:	yes, using trimpot

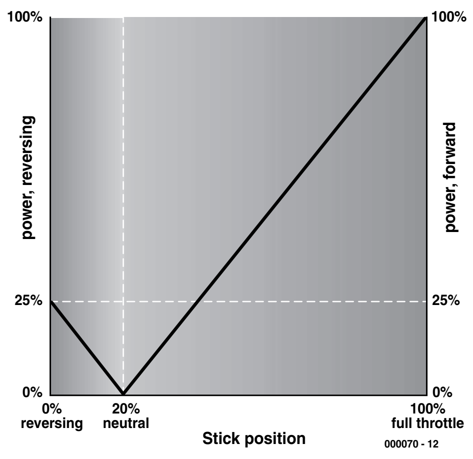


Figure 2. Three-point timing illustrated. The maximum forward and reverse power levels depend on the choice of the neutral position. The control characteristics are mirrored about a vertical line through the neutral point.

Three-point timing

'Three-point timing' forms an essential part of the SpeedControl concept. This has to do with how the SpeedControl is linked to the transmitter, and in particular, to three specific positions of the control stick.

Three-point timing is based on the following three positions of the control stick: reverse, neutral and full throttle. Neutral must lie between reverse and full throttle, but it may also be the same as reverse.

In neutral, the motor is always stopped. At full throttle, the motor is supplied with maximum power. If neutral is located beyond reverse, then when the stick is moved towards reverse, the 'for/back' output is first activated, and after that the power is again increased from 0%.

In the SpeedPower 1 unit, the 'for/back' signal is used to actuate the relay, while in the SpeedPower 2 unit it is used to control the brake FET and block power to the motor.

We can illustrate the versatility of this concept using a few examples. The SpeedControl unit is assumed to be connected to the receiver, with the transmitter switched on. The timing of reverse, neutral and full throttle takes place as follows. If the pushbutton of the SpeedControl is held down while power is switched on, the microcontroller will enter the 'timing' mode (under the condition that the trimmer potentiometer is set to 'time'), and the three LEDs 'max'. 'med' and 'min' will all be illuminated. The control stick can now be set to the 'reverse' position. If the pushbutton is pressed, the timing of the 'reverse' position is measured (as shown by the 'min' LED being illuminated). This takes a little while, since many measurements are taken and then averaged, so be sure to hold the control stick steady. After the beep, the control stick can be moved to the 'neutral' position. Pressing the pushbutton again causes the timing of the 'neutral' position to be measured (the 'med[ium]' LED is illuminated). Finally, the full throttle timing can be measured.

The three timing measurements must always be made in the sequence revere, neutral and full throttle.

If the control stick is at the bottom when the first measurement is made, that is the 'reverse' position. On the other hand, if it is at the top for the first measurement, then 'reverse' is at the top.

If the control stick is not moved between making the reverse and neutral timing measurements, there is no reverse (a small amount of hysteresis is programmed in for the transition to reverse).

On completion of the three-point timing measurements, the controller resets itself and enters the 'normal operation' mode. This mode begins – following an initialisation – with checking the relationship among the three stick positions. If this is invalid, the SpeedControl cannot go any further, and it lets this be known by loud beeping.

Example 1

Stick down is stop, stick up is full throttle.

First set the trim down, since it will not be used.

Select 'timing mode'. Move the stick to the bottom and measure the 'reverse' timing. Leave the stick at the bottom and measure the 'neutral' timing. Now move the stick to the top and measure the 'full throttle' timing. All done!

Example 2

Stick down is stop, stick up is full throttle. The trim is used to select forward/reverse, or to switch on the brake FET.

Select 'timing mode'. Set the trim down and move the stick to the bottom, and then measure the 'reverse' timing. Then set the trim to the top and measure the 'neutral' timing. Finally, move the stick to the top and measure the 'full throttle' timing. All done!

Example 3

Stick in the middle is stop, stick up is full throttle and stick down is reverse (or to switch on the brake FET).

First set the trim down, since it will not be used.

Select 'timing mode'. Move the stick to the bottom and measure the 'reverse' timing. Then move the stick to the middle and measure the 'neutral' timing. Finally, move the stick to the top and measure the 'full throttle' timing. All done!

Note:

Maximum power is always obtained in the greater part of the range of stick motion (see **Figure 2**).

- If the relationship between reverse, neutral and full throttle is invalid (which will be the case when the SpeedControl is switched on), all three timing LEDs ('min', 'med' and 'max') will be illuminated and the unit will beep furiously.
- If there are no pulses when the SpeedControl is switched on, the 'max' LED will be illuminated and there will be a fast beeping signal.
- If the control stick is not in the 'neutral' position when the SpeedControl is switched on, the 'med' LED will be illuminated and there will be a slow beeping signal.

During the three-point timing process, the pause intervals are measured in addition to the positions of the control stick. The pulse repetition intervals are computed from the measured times, and the minimum and maximum repetition intervals (which will normally be the same) are extracted from the measured times using a bubble sort. Small margins are applied to these times, which are subsequently used as limit values for judging the validity of the received signal. If the repetition interval (or rate) deviates from the measured and calculated value by more than ±3 %, the pulse is not allowed to pass, or during operation it is regarded as erroneous (this check can be disabled if desired). It is thus necessary to carry out the three-point timing process for each individual R/C installation.

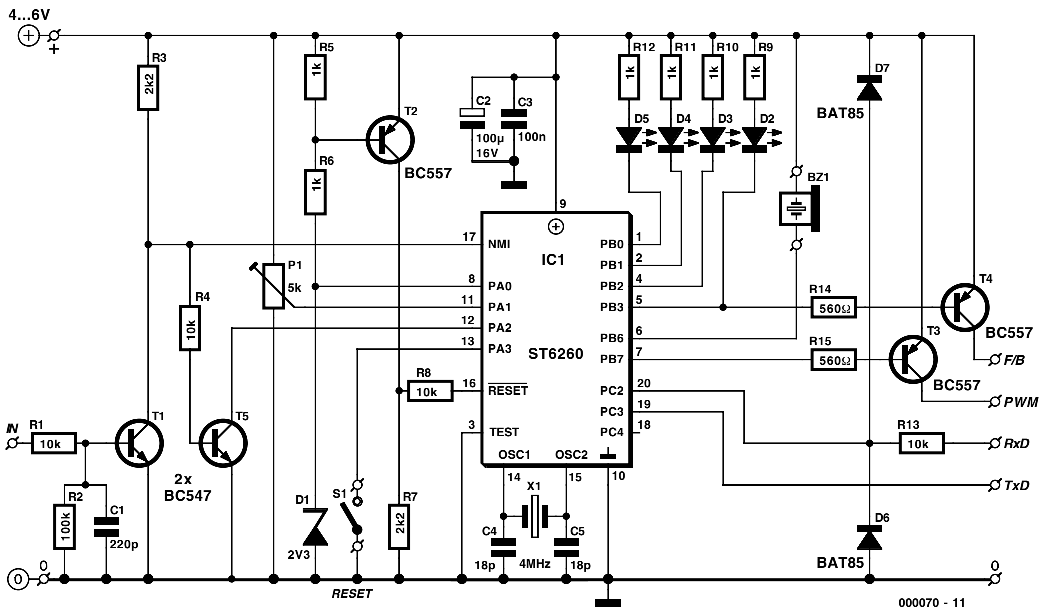


Figure 3. Schematic diagram of the SpeedControl unit.

The circuit

The schematic diagram of the SpeedControl is shown in **Figure 3**. The heart of the circuit is the ST62T60BB6, which is a powerful microcontroller with a large number of on-board peripheral circuits, such as a standard timer, a PWM timer, a digital watchdog, an A/D converter and a serial interface (SPI). There are 128 bytes of RAM present, as well as the same amount of EEPROM, along with 3384 bytes of program memory (user ROM/EEPROM). Although the maximum clock frequency is 8 MHz, we have here intentionally chosen 4 MHz, since the controller still works at this clock rate with a supply voltage of 3.5 V (instead of 4.5 V at 8 MHz).

The reset circuit, which consists of D1, T2 and R5–R8, deserves special attention. D1 is a 2.3-V voltage reference. The base of T2 is driven via the voltage divider R5/R6. The threshold voltage of T2 is around 0.6 V, so when the supply voltage reaches 4.2 V, T2 starts to conduct and the reset signal is disabled. This voltage lies well above the minimum operating voltage of the microcontroller, so we can be sure that it will never end up in an unstable region. The reference voltage is also connected to PA0. This port is configured as an input, and it can be switched over by the software to act as an analogue input. This allows the value of the supply voltage to be measured, in the interest of the cut-off function.

The RS232 interface is truly extremely simple: just two diodes (D6 and D7) and one resistor (R13). The input signal enters via port PC2, and the output signal exits via port PC3. PC2 is configured as the input to the serial interface (SPI) of the microcontroller, which handles incoming data on its own, without any intervention by the 'core' hardware. After reception is complete, the SPI interrupt sets a flag, and the core can fetch the data. The SPI is not used for transmitting RS232 data. A software routine for this purpose is present, and it drives PC3.

As it happens, the durations of the incoming servo pulses are measured using their rising and falling edges, each of which generates an interrupt. The NMI and the PA ports can detect falling edges, while the PC ports can detect rising edges. It would thus be really very nice if one edge could be captured with the NMI and the other edge with one of the PC ports. Unfortunately, this cannot be done, since the SPI occupies PC4 (as a clock line), so we have no choice but to use one of the PA ports. This means that the incoming signal must be inverted twice, first by T1 for buffering and level conversion (since the amplitude of the servo pulse is only 2.5 V with some receivers), and then again by T5 in order to generate an edge interrupt via PA2. The rising edge of the incoming pulse thus generates an NMI interrupt (which takes precedence over all other interrupts), while the falling edge generates a PA2 interrupt. The durations of the pulses and the pauses between pulses are measured by the built-in timer, which works pretty much on its own, under control of the interrupt routines.

The other inputs that are present are the trimpot P1, whose voltage can be read by the software, and pushbutton S1. If S1 is held down while the power is switched on, then one of three options is chosen, depending on the previously selected setting of the trimpot. These are 'time', 'program' and 'speed'. 'Time' puts us in the three-point timing mode, which means that after you have released the pushbutton you will have to press it three more times in succession (and wait for the beep) for the reverse, neutral and full throttle positions. In the 'program' mode, the controller can be programmed by the user entirely according to his or her own preferences, using a PC. In the 'speed' mode, no servo signal is necessary; the trimpot takes over the control function, as though it were a control stick.

All port B lines are configured as outputs. Each of them can sink or source up to 20 mA. The buzzer, the LEDs and the outputs to the power units ('PWM' and 'for/back') are connected to these lines (the latter two via open-collector drivers).

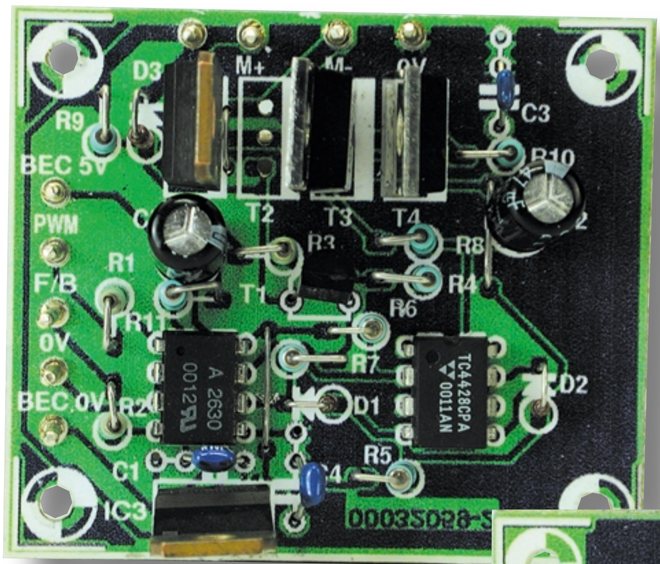
In addition to providing status information in

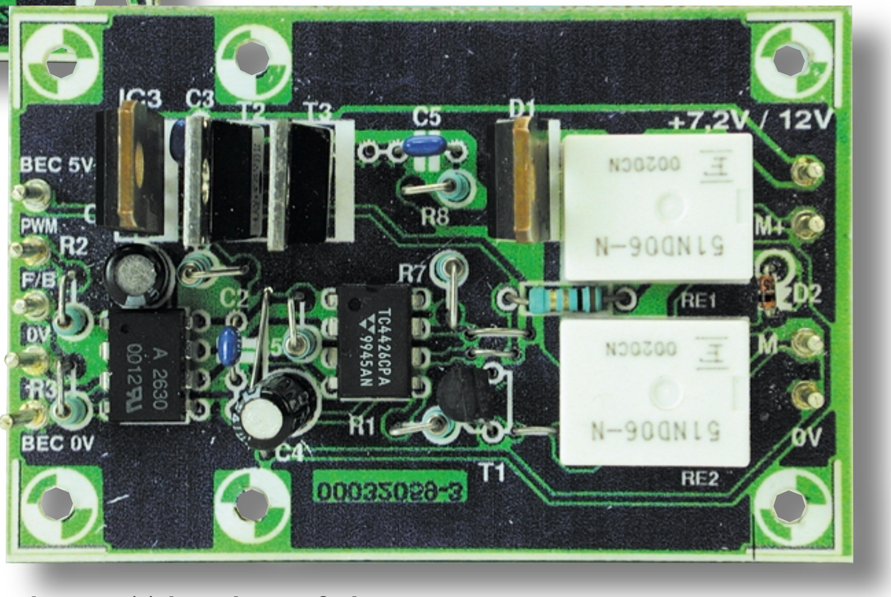
Pulse durations and cables

There is little in the way of standardisation to be seen in the area of radio control, but fortunately the differences are not that large, and nowadays a positive servo pulse is used nearly universally. The following table lists the pulse durations used by the best-known manufacturers, along with the repetition rates. The order numbers and colour codes for the servo cables are shown to the right.

Make	Pulse duration (ms)				Servo cable			
	min	neutral	max	Hz	order no.	+ batt	– batt	pulse
Futaba	0.9	1.5	2.1	50	F1439	red	black	white
Graupner/Jr	0.8	1.5	2.2	50	3941/6	red	brown	orange
Multiplex	1.05	1.6	2.15	40	890412	red	black	yellow
Robbe	0.65	1.3	1.95	50	8182	red	black	white
Simprop	1.2	1.7	2.2	50	0101745	red	blue	black

We used servo cables with Futaba plug/socket sets for the various connections, but you are naturally free to use whatever you wish. If you want to make up your own cables, bear in mind that genuine servo cables contain very fine stranded wire leads inside a silicone rubber mantle. The contacts are pressed into the leads, which yields more reliable connections than does soldering. If you solder your own connections, therefore, always finish them with sleeves of heat-shrink tubing.





special cases, the LEDs have the following significance in normal operation: the green LED is illuminated in the ‘no power’ state, the red LED D3 at ‘full throttle’ and the yellow LED for intermediate settings. The red LED D2 is connected to the ‘for/back’ output and is illuminated when this output is active.
In addition, the three ’power’ LEDs have an additional meaning if the user option ‘fail’ is enabled. In this case, these LEDs are illuminated if errors are detected in the incoming pulse stream (as long as the user has not disabled the ‘reptest’ check).
The final item to be noted in the schematic diagram is the PWM signal. The PWM timer (auto-reload timer) is not fed directly from the main program loop, but instead indirectly via a ‘sliding’ delay. This means that the outputs signal can never abruptly change from stopped to full throttle – this transition is always ‘sliding’. This reduces the strain on the motor(s), but what we find even more important is that this function enormously increases safety. Suppose that against all odds, a false pulse should manage to penetrate to the core of the software while the motor is stopped. If this pulse were to correspond to ‘full throttle’, the motor would give a tremendous jolt. Gliding motor control prevents this!
The PWM timer cannot generate outputs with 0% or 100% duty cycle, since spikes are always present in the output. The software thus switches PB7 over to operate as a normal output in such cases, and sets it to the appropriate state – with no spikes. The power MOSFETs in the power units appreciate the favour.

(000070-1)

Next month, we will continue with construction, programming and the power units.

Pulse Width Modulator

Using GAL16V8

Design by J. Hesse and R. Lessing

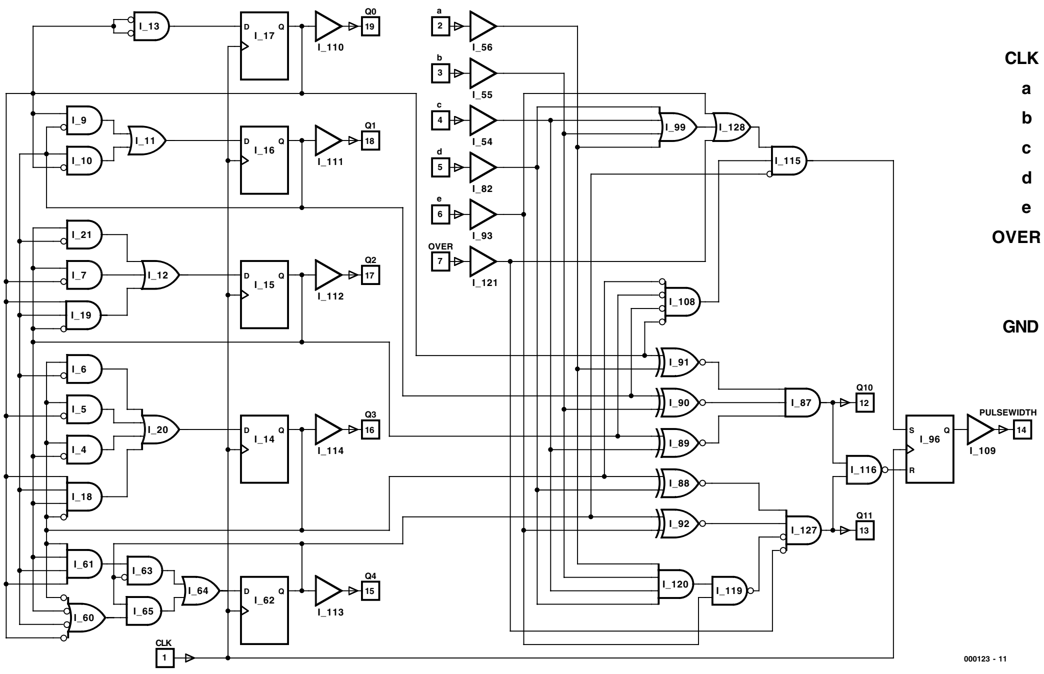
Pulse width modulation (PWM) is a fundamental technique in electronics. It comes as no surprise, therefore, that there are many applications for this digitally-programmed design.

A pulse width modulator can form the basis of many fascinating projects. The speed of a DC motor, for example, can be smoothly varied from zero to 100%. Connect a power MOSFET and you can control model trains, drills, lamps and many other devices. This design is particularly interesting because of its digital control input: operation is governed not by an analogue voltage, but rather by five binary inputs which can be connected to a PLC or a microprocessor.

A Question of Software

The only component in the pulse width modulator is a programmable logic device, type GAL16V8. This is ideal for small digital circuits and is available cheaply (around £2). Also, the development environment (the ABEL compiler) is free, and runs on an ordinary PC. A ‘digital’ pulse width modulator has the advantage over its analogue counterpart that it is practically insensitive to variations in supply voltage and requires no components such as potentiometers which are prone to wear. The cleanliness of the output signal can be verified on an oscilloscope.

Figure 1 shows the arrangement of logic inside the GAL. Pin numbers are shown in small squares. As can be seen, there are seven inputs (*a-e*, *OVER* and *CLK*), and



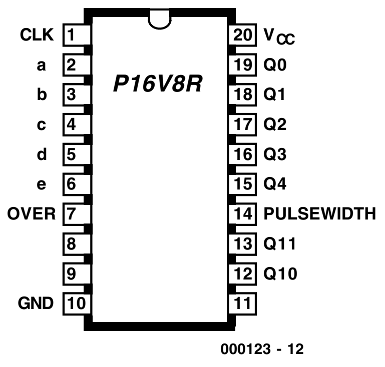


Figure 1. The logic circuit programmed into the GAL. The indicated inputs and outputs can be found on the device pinout.

six outputs (*Q0-Q4, Pulsewidth*) which are brought out to pins, as well as the internal signals *Q10* and *Q11*. *Q0-Q4*, although brought out to pins, are not used in the present application.

Pulses appearing on the *CLK* input are fed to a binary counter, which increments from 0 to 31. The counter consists of the logic gates on the left and five D-type flip-flops. The result at the output of the flip-flops is compared with a binary value presented at the inputs *a-e* and *OVER* (≥32). The comparator, which consists of the group of gates on the right-hand side of the circuit, drives the reset input of the final set/reset flip-flop. The set input is active when the counter is at zero. So, at the beginning of the count sequence the *Pulsewidth* output is set high, and it remains high until the programmed count is reached. Then the output flip-flop is reset and the output is low for the rest of the count.

The logic equations for the device (which are less complicated than they might appear!) are shown in **Figure 2**. They are written for the ABEL GAL compiler. The ABEL compiler is an elusive piece of software, but is provided as part of the *ispDesignEXPERT-Starter* package. This software, which can program PALs and GALs as well is ispLSI, MACH and ispGAL logic devices of up to 600 macrocells, is freely downloadable from the Lattice homepage http://www.latticesemi.com.

However, authorisation is required before the relevant files (more than 55 MB, plus manuals and tutorials) can be downloaded from http://www.latticesemi.com/lit/html/starter/ispde_starter.html and a licence number is also required for the software. This is all free, if rather tedious.

Any of the dozens of available versions of the GAL16V8 can be used. The suffixes simply indicate the maximum supply current and propagation delay (in ns). Suffix R has nothing to do with the type number: it indicates, in ABEL notation, that the GAL is used in 'registered' (as opposed to 'complex' or 'simple') mode. In registered mode, the clock input is always on pin 1 and output enable (OE) is always on pin 11. OE must be held low to enable the device's outputs. With other compilers, the suffix codes may be different.

PWM in practice

We now give some advice on using the pulse width modulator in practice. The clock frequency must be 32 times higher than the desired output switching frequency. For example, if a DC motor is to be driven, a clock frequency of at least 500 kHz is recommended. The resulting PWM frequency of 15.6 kHz is high enough to ensure that noise produced by the motor as a result of being driven with AC is of too high a frequency to be heard. If a higher frequency is used, eddy current losses will become too great and the motor may become too warm. (000123-1)

Software

The software consists of source code in ABEL-HDL, in the file PULS.REP or — if you would rather not get involved in ABEL code — as a JEDEC file PULS.JED for direct programming into the GAL. Both files are available on disk, order code **000123-11** or as a free download from the Elektor Electronics website www.elektor-electronics.co.uk.

```
P16V8R Programmed Logic:

Q10      = !(  !Q2.PIN & c
            #   Q2.PIN & !c
            #   !Q1.PIN & b
            #   Q1.PIN & !b
            #   !Q0.PIN & a
            #   Q0.PIN & !a );

Q11      = !(  over
            #   !Q3.PIN & d
            #   Q3.PIN & !d
            #   !Q4.PIN & e
            #   c & b & a & d & e
            #   Q4.PIN & !e );

Q3.D    = (  Q0.PIN & Q1.PIN & Q2.PIN & Q3.PIN
          #   !Q0.PIN & !Q3.PIN
          #   !Q1.PIN & !Q3.PIN
          #   !Q2.PIN & !Q3.PIN ); " ISTYPE 'INVERT'
Q3.C    = (  CLK );

Q2.D    = (  Q0.PIN & Q1.PIN & Q2.PIN
          #   !Q0.PIN & !Q2.PIN
          #   !Q1.PIN & !Q2.PIN ); " ISTYPE 'INVERT'
Q2.C    = (  CLK );

Q1.D    = (  Q0.PIN & Q1.PIN
          #   !Q0.PIN & !Q1.PIN ); " ISTYPE 'INVERT'
Q1.C    = (  CLK );

Q0.D    = (  Q0.PIN ); " ISTYPE 'INVERT'
Q0.C    = (  CLK );

Q4.D    = (  !Q4.PIN & !Q0.PIN
          #   !Q4.PIN & !Q1.PIN
          #   !Q4.PIN & !Q2.PIN
          #   Q4.PIN & Q0.PIN & Q1.PIN & Q2.PIN & Q3.PIN
          #   !Q4.PIN & !Q3.PIN ); " ISTYPE 'INVERT'
Q4.C    = (  CLK );

Pulsbreit.D    = (  Q11.PIN & Q10.PIN
                 #   Q4.PIN & Pulsbreit.Q
                 #   Q0.PIN & Pulsbreit.Q
                 #   Q1.PIN & Pulsbreit.Q
                 #   Q2.PIN & Pulsbreit.Q
                 #   Q3.PIN & Pulsbreit.Q
                 #   !over & !c & !b & !a & !d & !e & Pulsbreit.Q );
               " ISTYPE 'INVERT'
Pulsbreit.C    = (  CLK );
```

Notation: &= AND, , = OR, ! = negation

Figure 2. The logic expressed in ABEL-HDL.

Radio-Linked Caller Identification System

With up to 16 caller units

By Dr. Pei An pan@intec-group.co.uk

This article introduces a PC-based radio-linked caller identification system. It consists of a central receiver and up to 16 caller units. The receiver is connected to a computer via the Centronics port. Each unit has a push-to-make button on it. When the button is pressed, the caller unit sends out a unique code to the soundings. The receiver intercepts the code and the computer decodes it and displays the number of the activated caller unit on the screen. The operation distance is about 75 metres in buildings and 200 metres in open fields.

The system has a wide range of applications. In hotels, restaurant and shops, the caller unit can serve as a 'Call Assistance' button used by customers to call for assistance. It also has applications in interactive teaching environment in schools and colleges.

The system utilises '2nd generation' versions of FM radio transmitter and receiver modules (TX2/RX2) from Radiometrix Limited. The transmitter is a low-power device (LPD) type-approved to the Radiocommunications Authority (RA) specification MPT 1340 in the UK and Europe. This avoids the need to submit the project for final approval.

A Visual Basic 5 program has been developed for the system, demonstrating how the system is integrated with software.

How it works

Figure 1a shows the principle of the caller unit. Inside a caller unit (transmitter), an encoder IC (HT-12E) converts a 12-bit parallel data into serial data. The first eight bits represent a *system address* and the next 4 bits are the *address* of the caller unit. The system address ranges from 0 to 255 and is a value assigned to the system. The address of a call unit ranges from 0 to 15. The serial data is fed into the Radiometrix TX2 radio transmitter module, in which the serial data modulates a radio frequency carrier signal (at 418 MHz or 433 MHz, depending on the TX2 version used) using FM modulation. Next, the radio signal is transmitted to surroundings from an antenna.

The basic operation of the system receiver is illustrated in **Figure 2b**. Inside a receiver, an FM radio receiver module (RX2) demodulates the radio signal picked up by the antenna. Once demodulated, serial data is fed into a serial-to-parallel decoder IC (HT-12D), which converts the serial data back to parallel format. The address bits are compared with the preset address of the decoder. If they match, the 4-bit data is latched to the output (which is the address of the activated caller unit). If the address does not match, the decoder ignores the data. After a dataword is successfully received, the computer reads the data from the decoder and displays the number of the activated caller unit on the screen.

Radio transmitter and receiver modules

The Radiometrix radio transmitter and receiver modules (TX2 and RX2) allow a digital radio link to be implemented easily. The modules are a SAW-controlled FM radio transmitter and receiver which are specially designed for radio telemetry and tele-command applications (SAW = surface acoustic wave). There are a variety of TX2/RX2 modules that can be used with the present project. They are shown in **Table 1**.

A brief description of the transmitter and receiver modules is given here. Technical details of the modules can be found in the data

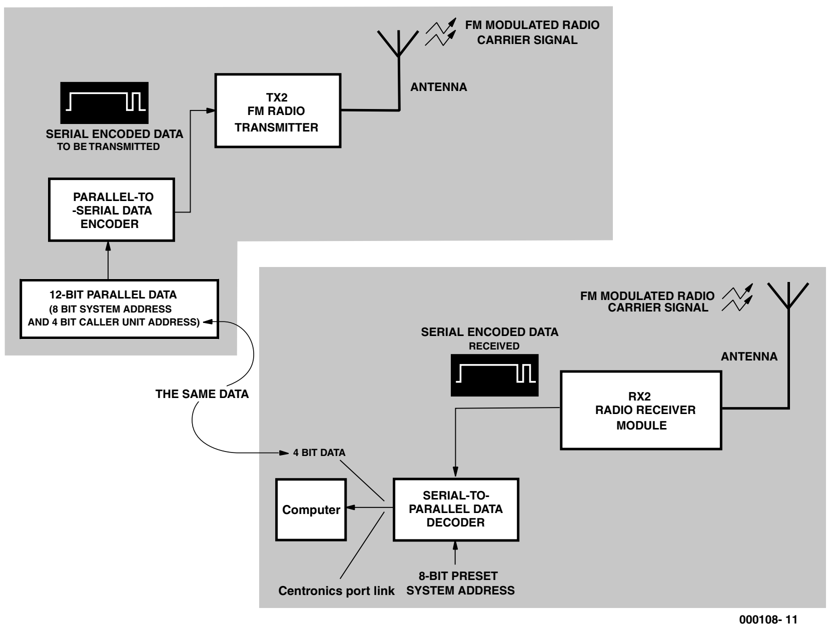

FM MODULATED RADIO CARRIER SIGNAL
ANTENNA
TX2 FM RADIO TRANSMITTER
SERIAL ENCODED DATA TO BE TRANSMITTED
PARALLEL-TO -SERIAL DATA ENCODER
12-BIT PARALLEL DATA (8 BIT SYSTEM ADDRESS AND 4 BIT CALLER UNIT ADDRESS)
THE SAME DATA
FM MODULATED RADIO CARRIER SIGNAL
ANTENNA
SERIAL ENCODED DATA RECEIVED
RX2 RADIO RECEIVER MODULE
4 BIT DATA
Computer
SERIAL-TO-PARALLEL DATA DECODER
8-BIT PRESET SYSTEM ADDRESS
Centronics port link
000108- 11

Figure 1. Principle of the Caller Identification system.

Table 1
Variant of TX2/RX2 radio link modules

Parameters	Description
Frequencies	418.00 MHz for UK use
	433.92 MHz for European use
Supply voltages	5V (4-6V for TX2 and RX2)
	3V (2.2V to 4V for TX2, 3 to 4V for RX2)
RX data transfer rate	-A: 7kHz baseband BW, slow data up to 14kbps
	-F: 20kHz baseband BW, fast data up to 40kbps

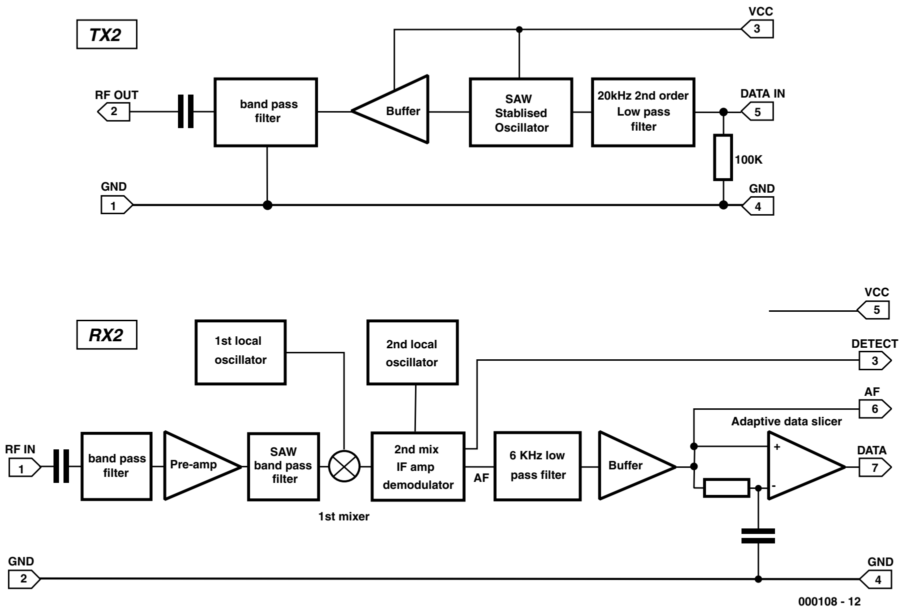
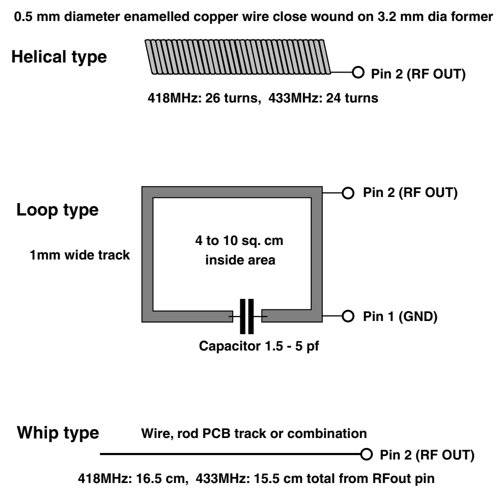


Figure 2. Internal block diagram of the radio link modules.


0.5 mm diameter enamelled copper wire close wound on 3.2 mm dia former
Helical type
Pin 2 (RF OUT)
418MHz: 26 turns, 433MHz: 24 turns
Loop type
1mm wide track
4 to 10 sq. cm inside area
Pin 2 (RF OUT)
Pin 1 (GND)
Capacitor 1.5 - 5 pf
Whip type
Wire, rod PCB track or combination
Pin 2 (RF OUT)
418MHz: 16.5 cm, 433MHz: 15.5 cm total from RFout pin

Antenna performance chart	Helical	Loop	Whip
Ultimate performance	√√	√	√√√
Ease of set-up	√√	√	√√√
Size	√√√	√√	√
Immunity to proximity de-tuning	√√	√√√	√

000108- 13

Figure 3. Antennas for the radio link modules.

sheets, Reference [1].

Transmitter Module type TX2
The pin functions of the transmitter are given in **Figure 2a**. For the +5 V and 433 MHz version, the operation voltage ranges from 4 to 6 V DC. The typical current consumption is about 10 mA at 5 V. For the +3 V and 433 MHz version, the operation voltage is between 2.2 V to 4 V DC with a typical current consumption of 6 mA at 3 V. Digital data to be sent (which should be a CMOS logic level at the same power supply voltage) is fed into pin 5. An antenna is connected to pin 2. The block diagram of the module is given in **Figure 2a**.

Receiver Module type RX2
The pin functions of the receiver are shown in **Figure 2b**. For the +5 V version, the operation voltage ranges from 4 to 6 V DC. The typical current consumption is about 13 mA at 5 V. For the +3 V version, the operation voltage is between 3.0 to 4.0 V DC with a typical current consumption of 13 mA at 3.5 V. The output digital signal appears at pin 7 (RXD) with a CMOS logic level. Pin 3 is the Carrier Detect output. It may be used to drive an external pnp transistor to obtain a logic level carrier detect signal. If not used, it should be connected to +5 V.

Antenna options
The antenna used with an LPD like the TX2 module is usually one of three versions: the helical type, the loop type and the whip type (see **Figure 3**). The helical antenna has a small size. It needs to be optimised for the exact wavelength in use. The loop antenna consists of a loop of PCB track, which is tuned by a variable capacitor. The quarter-wave whip-type antenna is a length of wire, a rod, a PCB track or combinations. **Figure 3** shows how the different versions are constructed and compares the performance amongst various antennas. The above

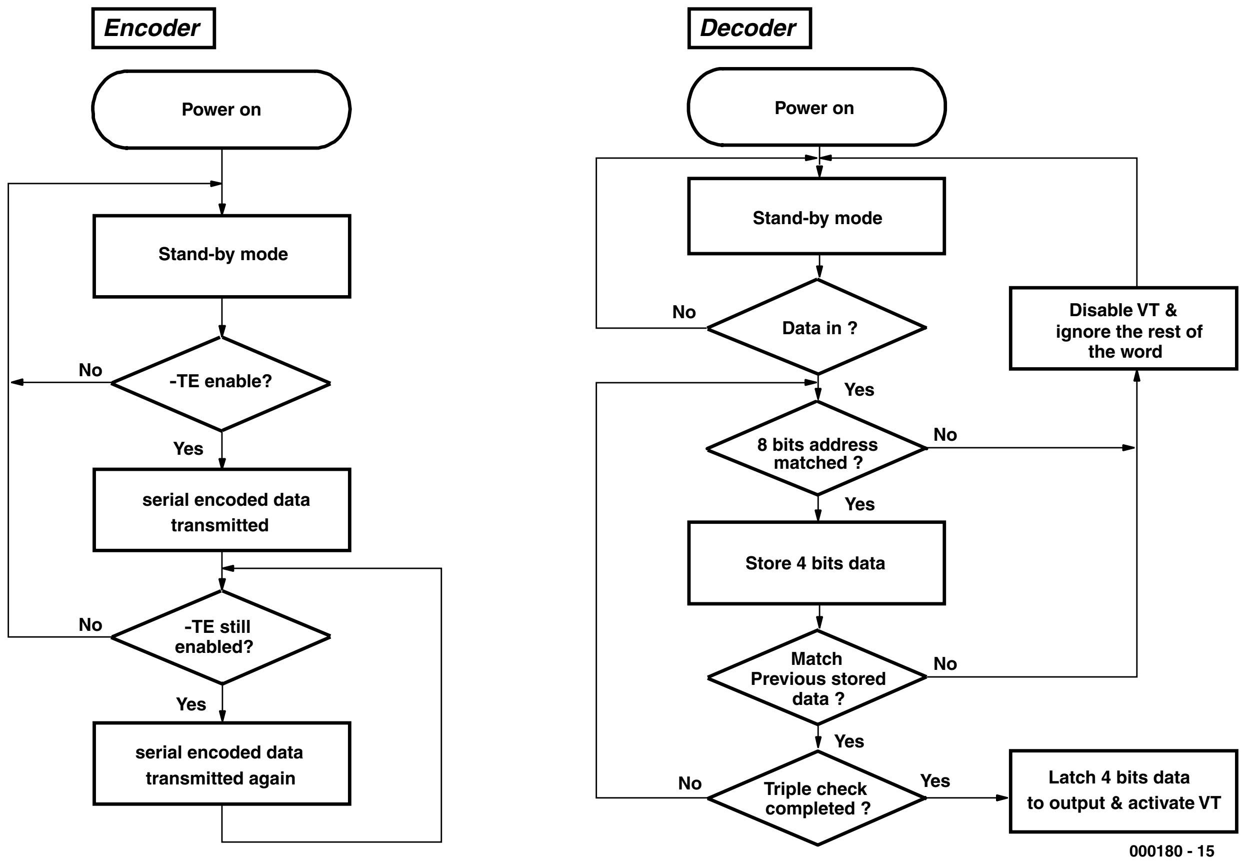


Figure 4. Flow charts of encoder and decoder operation.

antenna considerations also apply to the receiver.

Heart of the system: HT12-E/HT12-D

The HT-12E and HT-12D from Holtek are a CMOS encoder/decoder IC pair designed for digital code transmission and receiving. They have a wide operating voltage from 2.4 V up to 12 V with a typical stand-by current of 1 µA. They have an on-board oscillator that requires one external resistor. Brief descriptions of the ICs are given here. For more details, please refer to the manufacturer's datasheet, Reference [2]. When pin numbers are mentioned below, you may already refer to the circuit diagrams. The operation of these interesting ICs is conveniently described using flowcharts, see **Figures 4a** and **4b**.

The **HT-12E encoder** encodes 12-bit of parallel data into a serial data. It transmits the data upon the receipt of a falling edge at the Transmit Enable pin ($\overline{\text{TE}}$). The 12 bits of data consist of eight bits of address (A0-A7 connected to pins 1-8) and four bits of data (D3-D0) connected to pins 10-13). Therefore the total number of address combinations is 2^8 or 256. The external oscillator resistor is connected between pins 15 and 16. The serial data is output from pin 17.

The operation of the HT-12E is that initially the encoder is in the stand-by mode. Upon receipt of a $\overline{\text{TE}}$ signal (active Low), it begins a 4-word transmission cycle and repeats the cycle until the $\overline{\text{TE}}$ signal becomes high (see Figure 4a). Each word contains two periods: the pilot code period and code periods as shown in **Figure 5a**. The pilot code period has a 12-bit length period and is at logic Low. The code period also has a 12-bit length period and contains the serial encoded data. The encoder detects the logic state of the 12 bit inputs (A0-A7 and D0-D3) and encodes the information into a serial datastream. The logic levels '0' and '1' are encoded as shown in **Figure 5b**. The order of the databit transmission is from A0 to A7, then from D3 to D0.

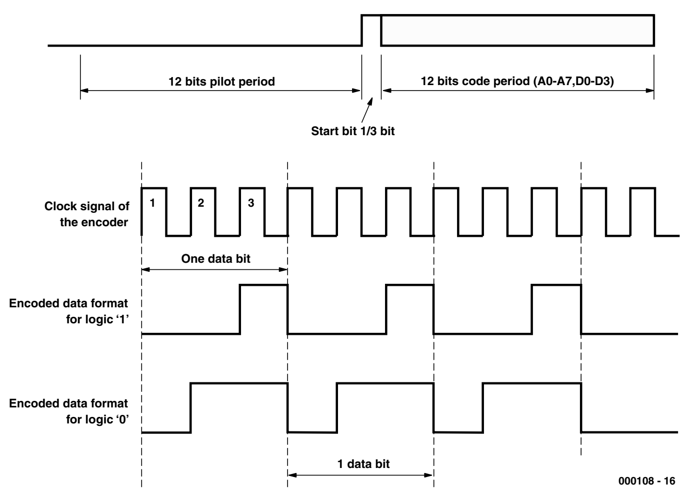

12 bits pilot period
12 bits code period (A0-A7,D0-D3)
Start bit 1/3 bit
Clock signal of the encoder
1
2
3
One data bit
Encoded data format for logic '1'
Encoded data format for logic '0'
1 data bit
000108 - 16

Figure 5. Data format of one transmission (a) and encoded serial data format (b).

The **HT-12D decoder** receives the 12-bit word and interprets the first eight bits as the address and the last four bits as data. When the received address matches the decoder's pre-defined address, the Valid Transmission (VT) output goes High and the 4-bit data is latched to the data output pins. A triple check scheme is used to enhance the reliability of data reception (Figure 4b). The VT output remains high until the right code is not received anymore. The pre-defined address for

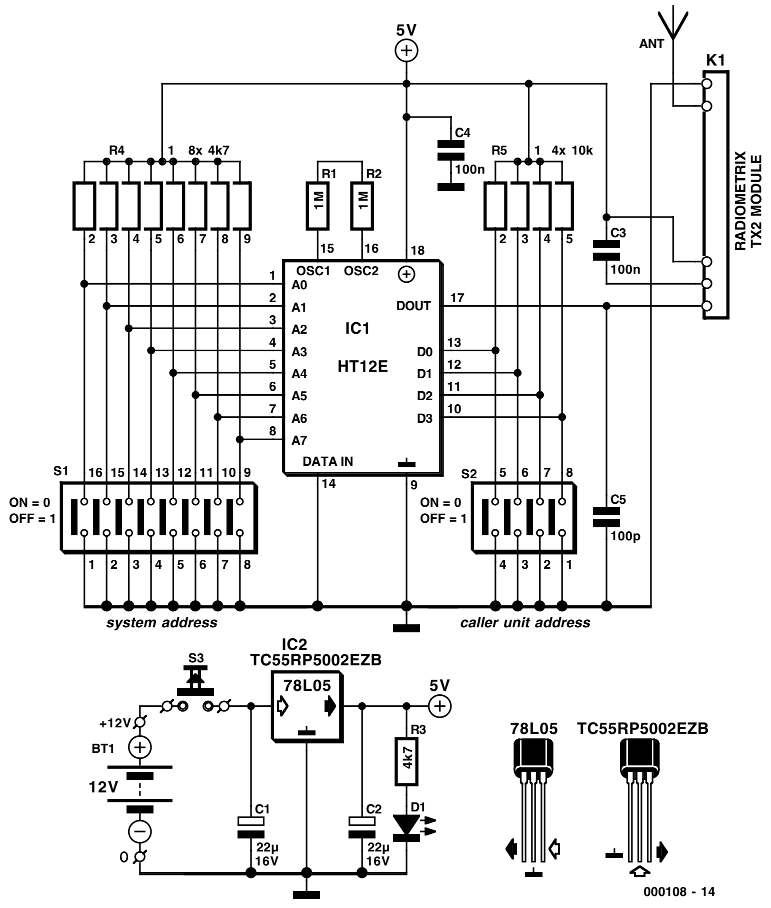


Figure 6. Circuit diagram of the caller unit. Up to 16 of these may be used within a single system address.

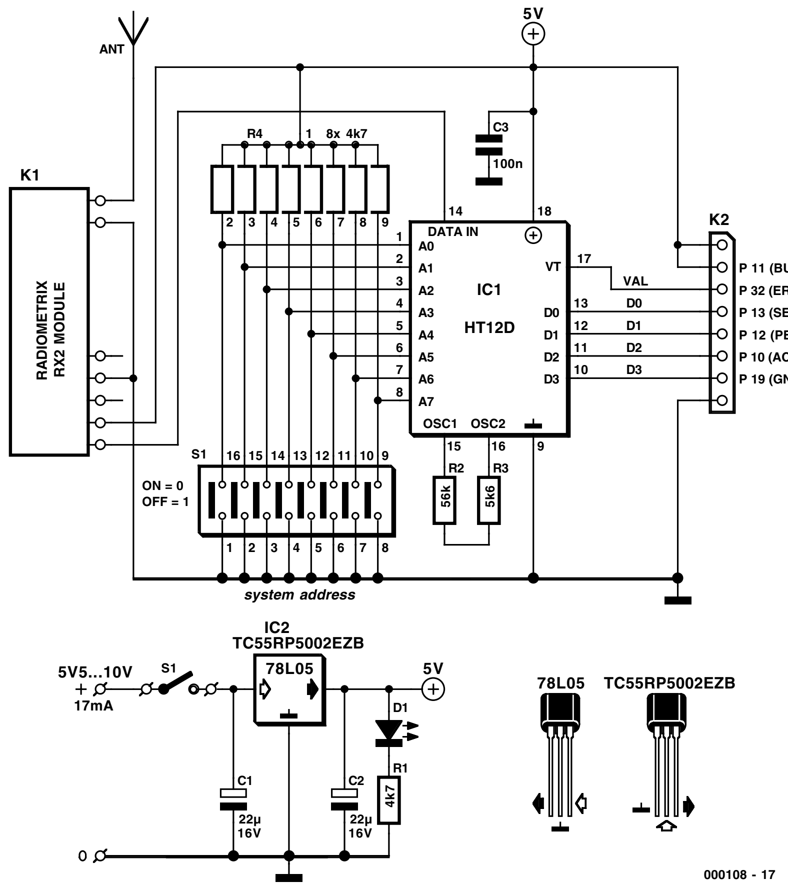


Figure 7. Circuit diagram of the central receiver unit.

Table 2. Some values of external resistors for encoder and decoder.

HT-12E (encoder)		HT-12D (decoder)	
R	F_{osc}	R	F_{osc}
1.1 M	3 kHz	62 k	150 kHz
750 k	4.3 kHz	33 k	240 kHz

the encoder is determined by the logic states at pins 1-8 for A0-A7. The latched data is output from pins 10 to 13. The serial data is input at pin 14. The external oscillator resistor is connected between pins 15 and 16. Pin 17 is the valid transmission (VT) indicator output. The external resistors required on the OSC pins of the HT12D/E are 1% types. **Table 2** gives the resistance values for 3 kHz and 4.3 kHz clock frequencies, F_{OSC} (in the present circuit, F_{OSC} = 3 kHz). For other frequencies, please refer to the manufacturer's data sheet.

Circuit of caller unit

Figure 6 shows the circuit diagram of the transmitter. The encoder, HT-12E, converts an 8-bit address (system address set by DIP switch block S1) and a 4-bit data (caller address set by DIP switch block S2) into a serial data form. The serial data is output from the Dout pin, number 17. The $\overline{TE}$ input (Transmit Enable, pin 14) is set permanently to Low to enable data transmission.

The serial data generated by the HT-12E encoder is fed into pin 5 of the TX2 radio transmitter module which is plugged or soldered onto connector K1. The transmitter signal emerges from pin 2 of the TX2 module and is radiated by an antenna.

The power supply to the caller unit is an N-type 12-V alkaline battery (10 mm diameter and 28 mm length, 33 mAh capacity). The voltage is regulated to +5 V by a low-power and low-drop voltage regulator type TC55RP5002EZB from Telcom. Alternatively, an 78L05 may be used, but unlike the Telcom IC it will not allow the battery voltage to drop to 5.5 V before the transmitter becomes inoperative. Also note the different pinouts of the two regulators! S3 is a push button switch (push to make type). If the button is not pressed, no voltage is supplied to the circuit. If it is pressed, the caller unit immediately transmits the code. This arrangement allows a maximum battery power saving to be achieved. During a transmission, current consumption is about 15 mA. A calculation indicates that an N-type battery will last 2.3 hours of data transmission. Assuming that the activation time of switch is 1 second on average, the battery will last for at least 2,300 calls.

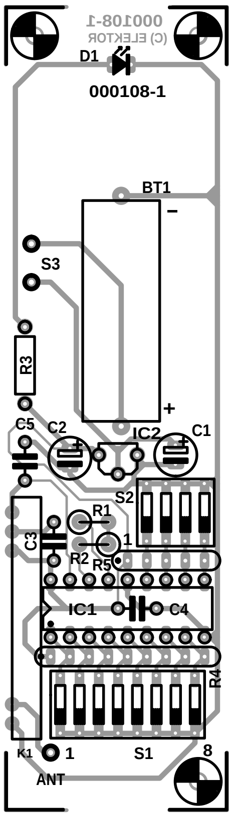
COMPONENTS LIST
Caller Unit (TX)

Resistors:
R1,R2 = 1MΩ 1%
R3 = 4kΩ7
R4 = SIL array, 8-way, 10kΩ
R5 = SIL array, 4-way, 10kΩ

Capacitors:
C1,C2 = 22µF 16V radial
C3,C4 = 100nF, 5mm lead pitch
C5 = 100pF

Semiconductors:
D1 = LED, high efficiency
IC1 = HT-12E (Holtek), RS Components # 854-100, Maplin # AE17T, Conrad # 175420
IC2 = TC55RP5002EZB, RS Components # 207-0045

Miscellaneous:
K1 = Transmitter module, RadioMetrix type TX2, Farnell # 722-4930 (418 MHz) or 722-4953 (433 MHz)*
S1 = 8-way DIP switch
S2 = 4-way DIP switch
S3 = pushbutton, 1 make contact
Antenna, see text
BT1 = 12V N-size battery (L1028)
PCB, order code **000108-1** (see Readers Services page)
ABS case, size 124 x 33 x 30 mm, Maplin # FT31J



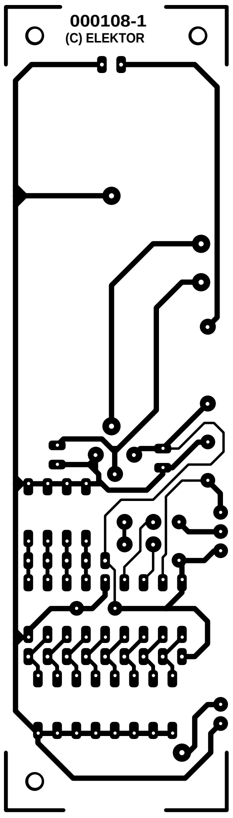


Figure 8. Track layout and component mounting plan of caller PCB (board available ready-made).

Circuit of receiver unit

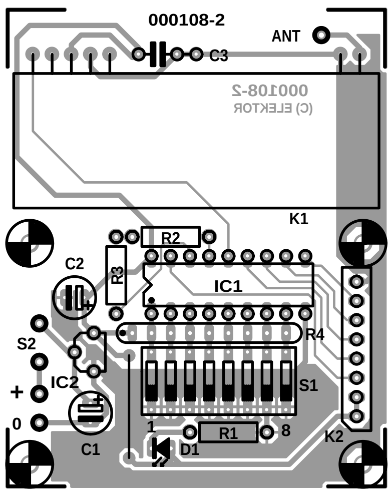
Figure 7 gives the circuit diagram of the central receiver. The radio frequency signal is picked up by an antenna and is fed into pin 1 of the RX2 receiver module. The demodulated signal is output from pin 7 and is fed into the decoder IC via pin 7 of connector K1.

The HT-12D decoder chip receives the serial data at pin 14, checks for errors and outputs the received data if it is a valid transmission. The first eight bits are treated as address bits and must match the pre-set address of the decoder (set by DIP switch array S1). If the two addresses match, the next four data bits are compared with the last data stored. If the data matches three times, the Valid Transmission output (pin 17) will go high. If not, VT remains low.

The supply voltage is regulated to +5 V by a low-power and low-drop voltage regulator, TC55RP500. Alternatively, a 78L05 may be used.

The printer status lines of a Centronics port are used to read data into the computer. The four data outputs from the HT-12D (D0-D3) and the VT (Valid Transmission) output are connected to the five input lines (bits 3-7) of the status port. More details on the Centronics port and its applications may be found in Reference [3].

Because the receiver has to be on all the time to receive calls from caller units, it is



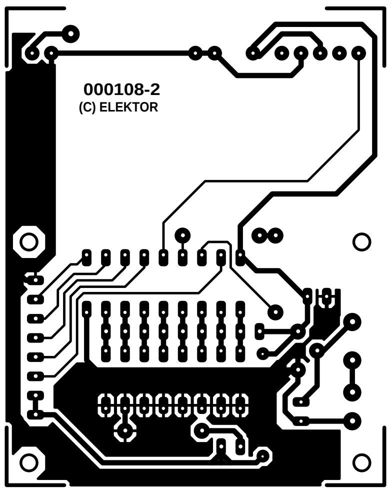


Figure 9. Track layout and component mounting plan of receiver PCB (board available ready-made).

COMPONENTS LIST

Receiver Unit (RX)

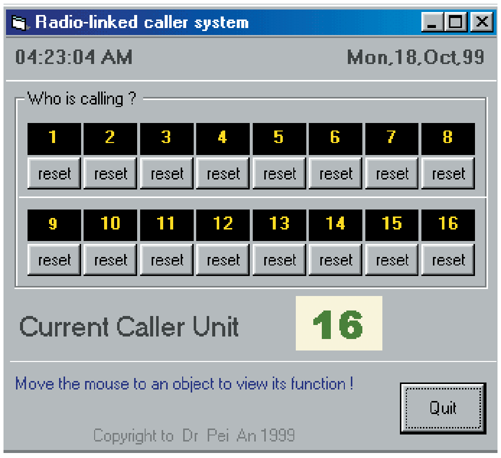
Resistors:
R1 = 4kΩ7
R2 = 56kΩ 1%
R3 = 5kΩ6 1%
R4 = 8-way SIL array, 4kΩ7

Capacitors:
C1,C2 = 22µF 16V radial
C3 = 100nF

Semiconductors:
D1 = LED, high efficiency
IC1 = HT-12D (Holtek), RS Components # 854-116, Maplin # AE18U, Conrad # 174351
IC2 = TC55RP5002EZB, RS Components # 207-0045

Miscellaneous:
K1 = Receiver module, RadioMetrix type RX2, Farnell # 722-4941 (418 MHz) or 722-4965 (433 MHz)*
K2 = 8-way SIL pinheader
S1 = 8-way DIP switch
S2 = on/off switch
Antenna, see text
ABS case, size 125 x 62 x 38 mm, Maplin # BZ73Q
PCB, order code **000108-2** (see Readers Services page)
Disk, project software, order code **00108-11** (see Readers Services page)

* Radiometrix distributor for the UK: Low Power Radio Solutions (LPRS), http://www.lprs.co.uk


Radio-linked caller system

04:23:04 AM Mon,18,Oct,99

Who is calling ?

Current Caller Unit 16

Move the mouse to an object to view its function !

Copyright to Dr Pei An 1999

Figure 10. Screendump of the 'Call Centre' software which runs under Windows.

powered by a small mains adaptor with an unregulated output voltage between about 6 V and 10 VDC.

Construction

The caller and receiver are constructed on single-sided PCB boards (**Figures 8 and 9**).

The helical antenna of the receiver is constructed using a piece of copper wire (see Figure 3 for construction details).

The parts lists suggest plastic (ABS) cases that may be used for the transmitter and receiver units. These cases pass RF signals at 70 cms and so allow the antennas to be incorporated. If the boards are mounted in metal cases, the antennas should not be enclosed. In most cases, that will leave the quarter-wave whip as the only viable option.

Construction of the caller and receiver units is rather simple. After soldering all the components on the PCB boards properly, and setting the DIP switches, the system will work straight away. Actually, the only thing you have to be careful with is the mounting of the voltage regulators as the 78L05 and TC55RP500 have **different pinouts**. There is no adjustment needed for the caller units and the receiver. Photographs of our prototypes are shown in **Figures 11 and 12**.

Software driver

The receiver is coupled to a programs that runs on a PC. The software driver is written in Visual BASIC 5. Readers should have some degree of knowledge about the Centronics port in order to understand the following description of the system software.

Bit 3 through to bit 7 of the Status port read data from the decoder. Bit 7 is connected to VT output of the decoder. When no transmission is received, VT remains low. When one of the caller units is activated and a valid transmission is received, VT ramps high. It remains high until the caller unit is no longer activated. So by polling this line at a regular interval, the computer will know if a caller unit is activated. Then, the computer reads data A0-A3 through bits 3-6 of the Status port. A0-A3 are the address bits of the most recently activated caller unit.

The operation of the software driver is illustrated in **Figure 10**. On the screen, there are 16 boxes corre-

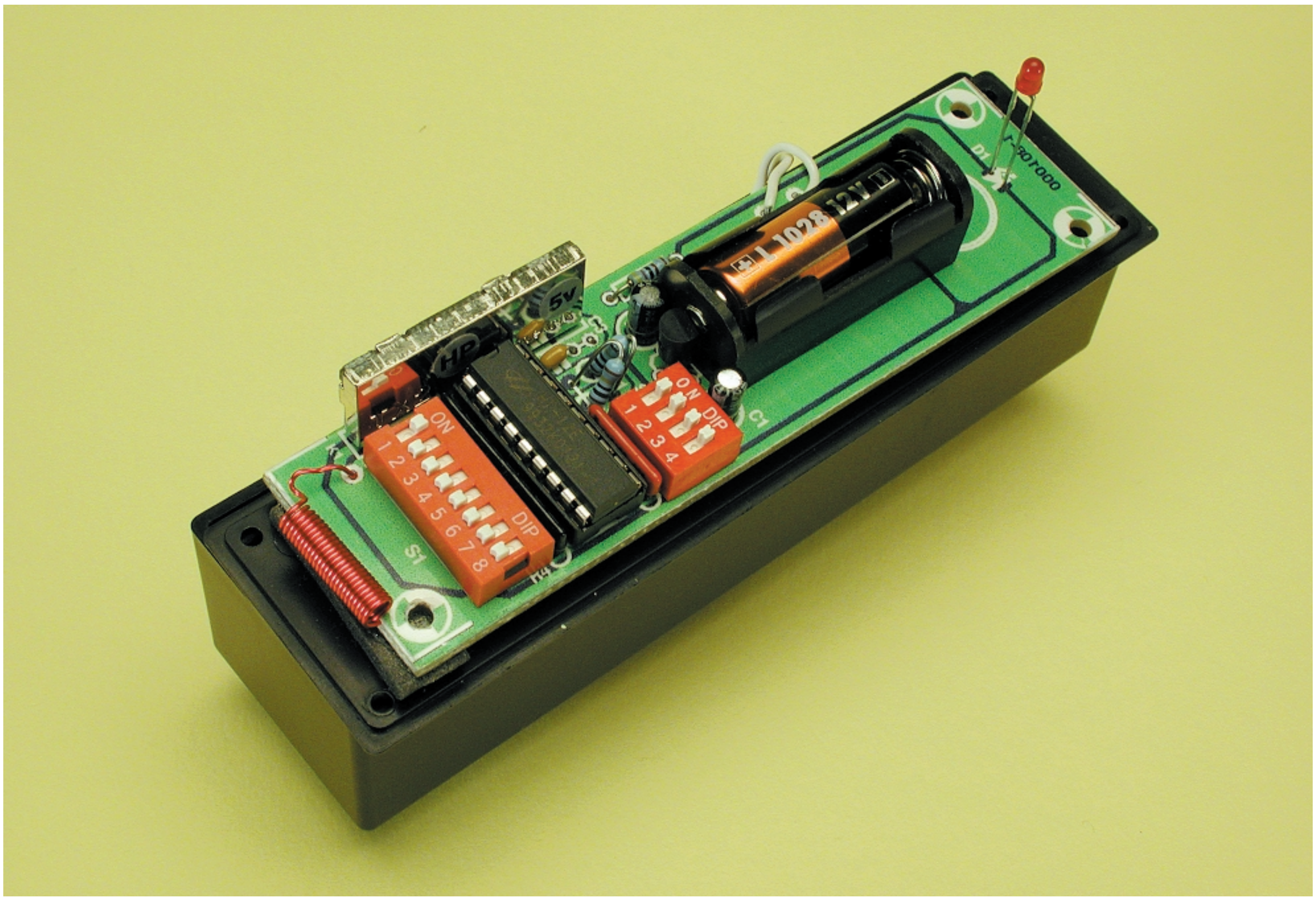


Figure 11. Completed caller board.

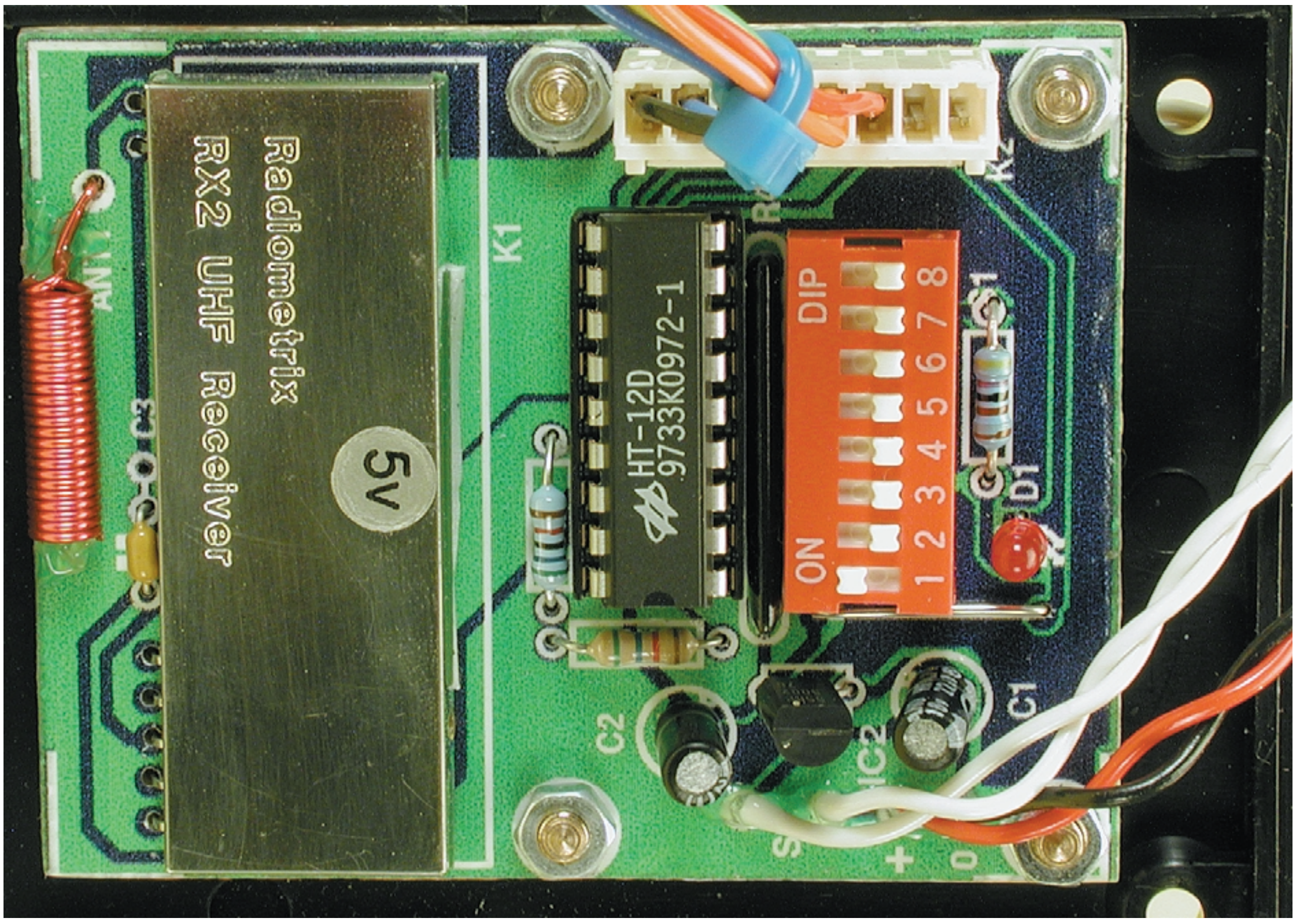


Figure 12. Completed receiver board.

sponding to Caller unit 1 to Caller unit 16. If a caller unit has never been activated, the box is black. If the caller unit is activated, the box becomes red to indicate that there is a call from that caller unit. Click the 'reset' button below the box to clear the call (box becomes black). The address of the most recently called unit is also shown on the screen.

It is known that VB5 is unable to read data directly from a hardware port. Therefore, a DLL I/O function is used. The DLL library is called `iosys.dll` and is written in Visual C. The I/O function (`inport`) should be declared in the `Callersys.bas` module first, then it can be called elsewhere in the program. This DLL, along with all software for the project, is available on a diskette set, order code **000108-11.** All project software is also available from the Free Downloads section of our website at http://www.elektor-electronics.co.uk. Look for item # 000108-11 under January 2001.

(000108-1)

Acknowledgement

The author expresses his gratitude for the help he received from Mr. Kangeyan at Radiometrix Ltd.

References

1. Datasheets for TX2 and RX2, available from Radiometrix Ltd. Tel. (+44) (0)1814 281220. Website: *http://www.radiometrix.co.uk*
2. Data sheets for HT-12 series from Holtek. Website: *http://www.holtek.com.tw*
3. PC Interfacing - Using Centronics, RS232 and game ports, Pei An, Newnes, Butterworth-Heinemann, 1998, ISBN 024 0514 483

On Project disk set 000183-11a/b/c/

Diskette 000108-11a	**Diskette 000108-11b**	**Diskette 000108-11c**
AsycFilt.dl1	AsycFilt.dl2	Callsys.ex_
Contents.txt	ComCat.dl_	Contents.txt
Copyright.txt	Contents.txt	Copyright.txt
MSVBVM50.dl_	Copyright.txt	Dao350.dl_
OlePro32.dl_	Ctl3d32.dl_	MSRD2x35.dl_
Setup.exe	iosys.dl_	MsRepl35.dl_
Setup.lst	MSJet35.dl_	ODBCJI32.dl_
setup1.ex_	MSJInt35.dl_	ODBCJt32.dl_
St5unst.ex_	MSJtEr35.dl_	ODBCTL32.dl_
StdOle2.tl_	MSVCRT40.dl_	OleAut32.dl_
VB5StKit.dl_	VB5DB.dl1	VB5DB.dl2
	VBAJet32.dl_	

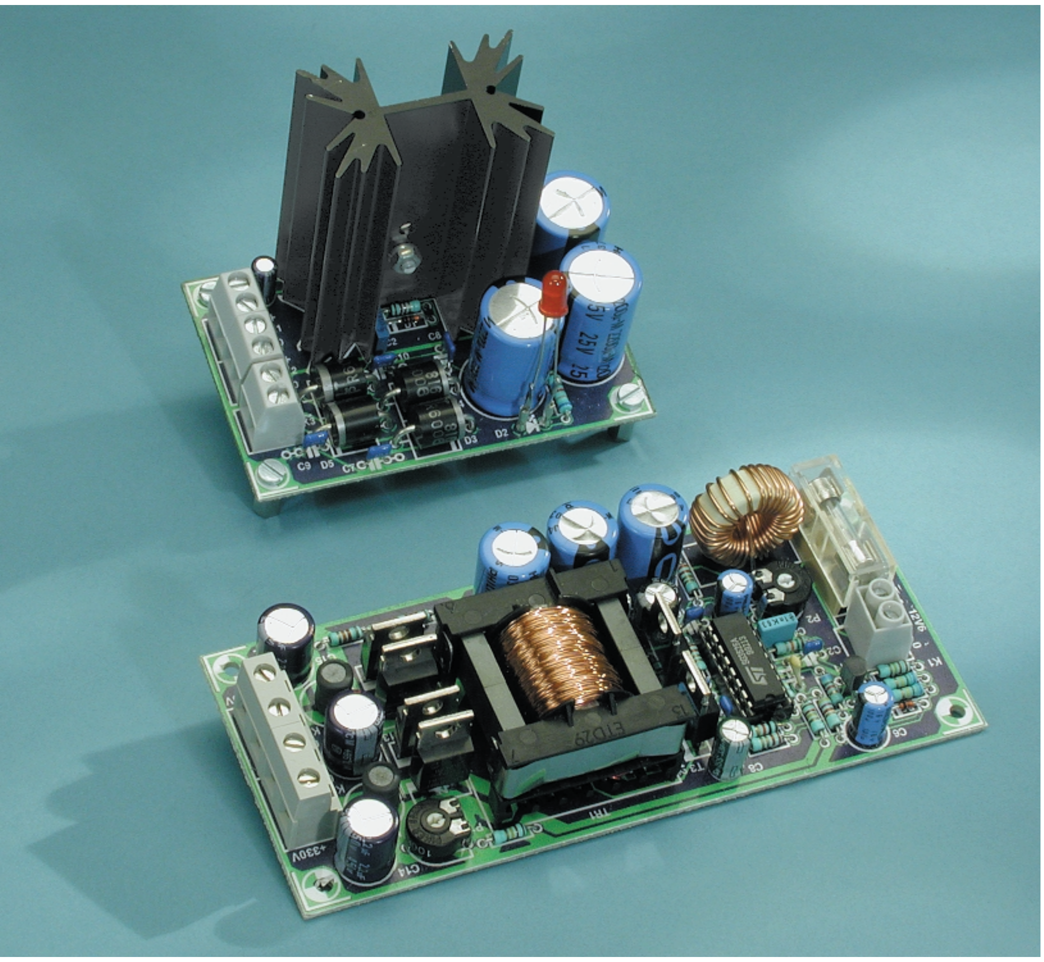
For the Valved RIAA Preamplifier and other applications

High Voltage Supply

330 V from 12 V

Design by T. Giesberts

Although this supply was primarily designed for use with the Valved RIAA Preamplifier, we found that the inverter stage is useful in many other applications. With only a small modification this circuit can be used to power a 20 W PLCE (low energy) lamp from a 12 V car battery.



The Valved RIAA Preamplifier uses two valves, just like the Valve Preamplifier that was published in the June 2000 edition of *Elektor Electronics*. Since the valves' filaments have again been connected in series, the preamplifier requires two DC supply voltages: 12.6 V for the filaments and 330 V for the high voltage supply.

In order to avoid the need to use a custom transformer the circuit has been designed to use a standard 15 V/3 A mains transformer. As we'll see later, the supply circuit consists of two distinct sections: a conventional 12.6 V filament supply and a step-up converter which boosts the 12.6 V to 330 V. In other words, the filament supply is also used to power the inverter.

Since each section is built on a separate PCB it becomes possible to use them individually in other applications. This is especially useful in case of the inverter, since it makes a great camping light when used in conjunction with a 12 V car battery and a PLCE lamp. These lamps tend to work very well off a 300 V DC supply!

The 12.6 V supply

As we can see in **Figure 1**, this section is a very basic circuit. The 3 A fixed-voltage regulator (TO-220 case) is made to deliver a slightly higher output (12.6 V) by adding an extra diode (D1). The bridge rectifier uses 6 A diodes and is followed by some substantial smoothing capacitors (C4, C5, C6). The bridge rectifier is RF decoupled by C7-C10 and LED D2 functions as the power indicator.

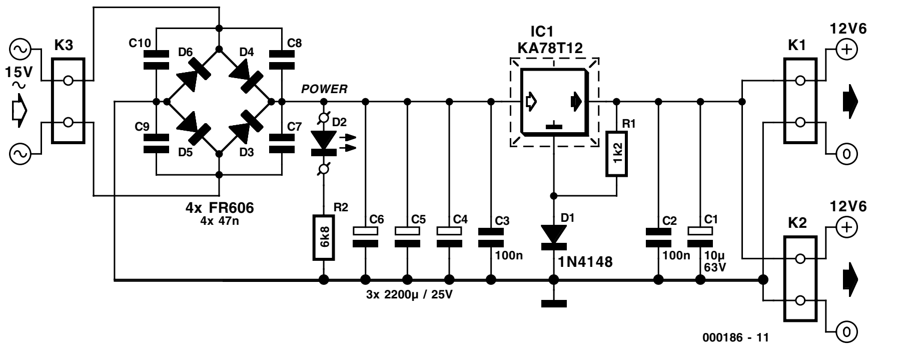


Figure 1. The 12.6 V supply incorporates the well-known diode 'trick', which causes the output to increase by 0.6 V.

Construction of the 12.6 V supply shouldn't cause any problems when the PCB shown in **Figure 2** is used. The heatsink for IC1 (Fischer type SK129 from Dau Components) is placed directly onto the PCB, which results in a compact module. It is very important that an insulating washer is used between IC1 and the heatsink.

There are two PCB terminal blocks (K1, K2) that provide the 12.6 V output voltage. One of these supplies the inverter and the other powers the two in series connected filaments. The third terminal block (K3) is for the 15 V transformer, which should be rated at least 50 VA.

The 330 V inverter

This part of the supply (see **Figure 3**) is a push-pull-converter that uses an old favourite of ours: the SG3525A. This regulator is an industry standard part that is used in many switch mode supplies. We have used it before in the 'In-Car Audio Amplifier', which we published in 1994. The proper description of this IC is a 'regulating pulse width modulator', which sums up its function perfectly. A special transformer is driven with an alternating voltage by one or more switched transistors, with the driving voltage obviously limited to a safe value. By varying the pulse width of the signal, the amount of power is controlled. The output at the secondary of the transformer is rectified and fed back to the PWM regulator in order to keep the output stable. That completes the feedback loop of the regulator.Since the SG3525A has been described in depth before in Elektor, we will limit ourselves to a brief overview of the device. The regulator uses a reference voltage of 5.1 V. Various internal circuits use this reference: error amplifier, oscillator, PWM comparator and the current source for the soft start. An extra delay circuit has been added to give valve amplifiers enough time to warm up before the HV supply is applied. Because valve amplifiers generally have substantial smoothing capacitors, the soft start period has been increased and the value of 100 µF for C5 is a fair bit higher than usual.

COMPONENTS LIST

12.6 V Supply

Resistors:
R1 = 1kΩ2
R2 = 6kΩ8

Capacitors:
C1 = 10µF 63V radial
C2,C3 = 100nF
C4,C5,C6 = 2200µF 25V radial
C7-C10 = 47nF ceramic

Semiconductors:
D1 = 1N4148
D2 = red high-efficiency LED
D3-D6 = FR606 (or similar 6A diode)
IC1 = KA78T12 (3A)

Miscellaneous:
K1,K2,K3 = 2-way PCB terminal block, lead pitch 5mm
For IC1: heatsink type SK129 63,5 STS, 3.5 K/W (Fischer) (Dau Components)
Isolation material for IC1
PCB , order code **000186-2** (see Readers Services page)

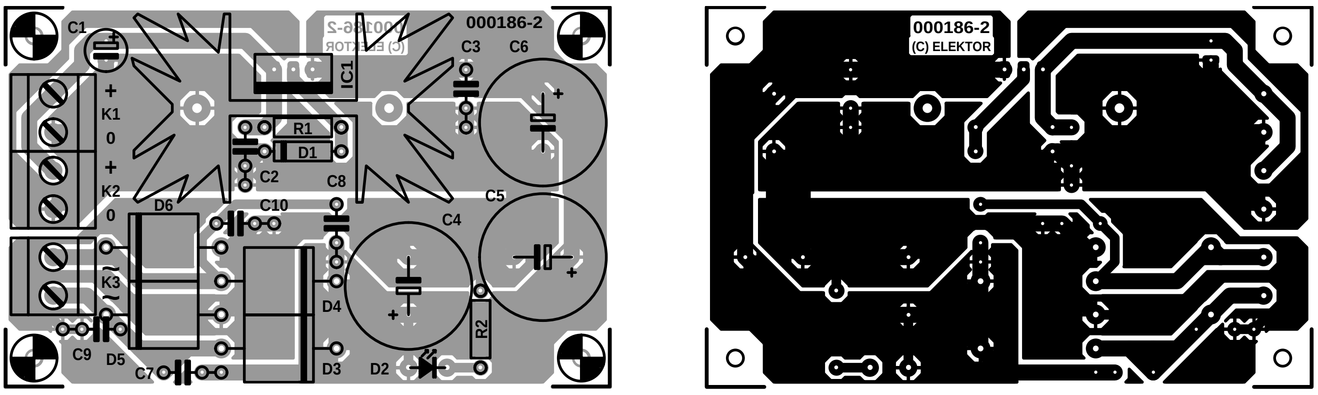


Figure 2. The heatsink just fits on the PCB, which results in a nice compact 12.6 V module.

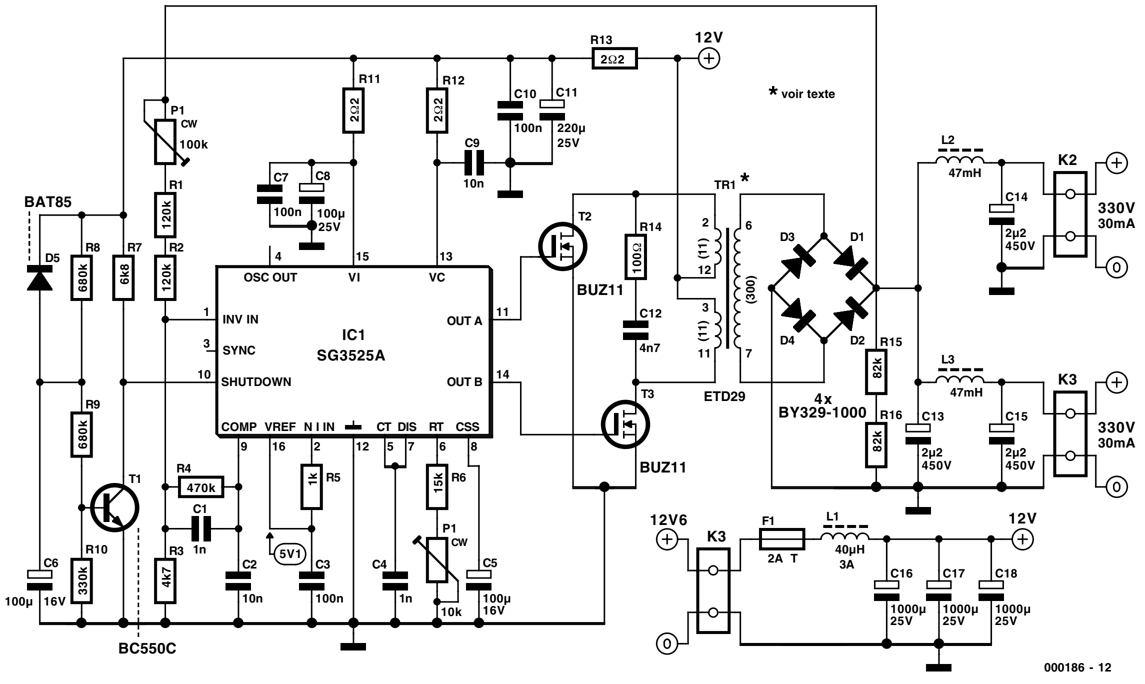


Figure 3. The main parts of the inverter are the integrated regulator (IC1), transformer and bridge rectifier.

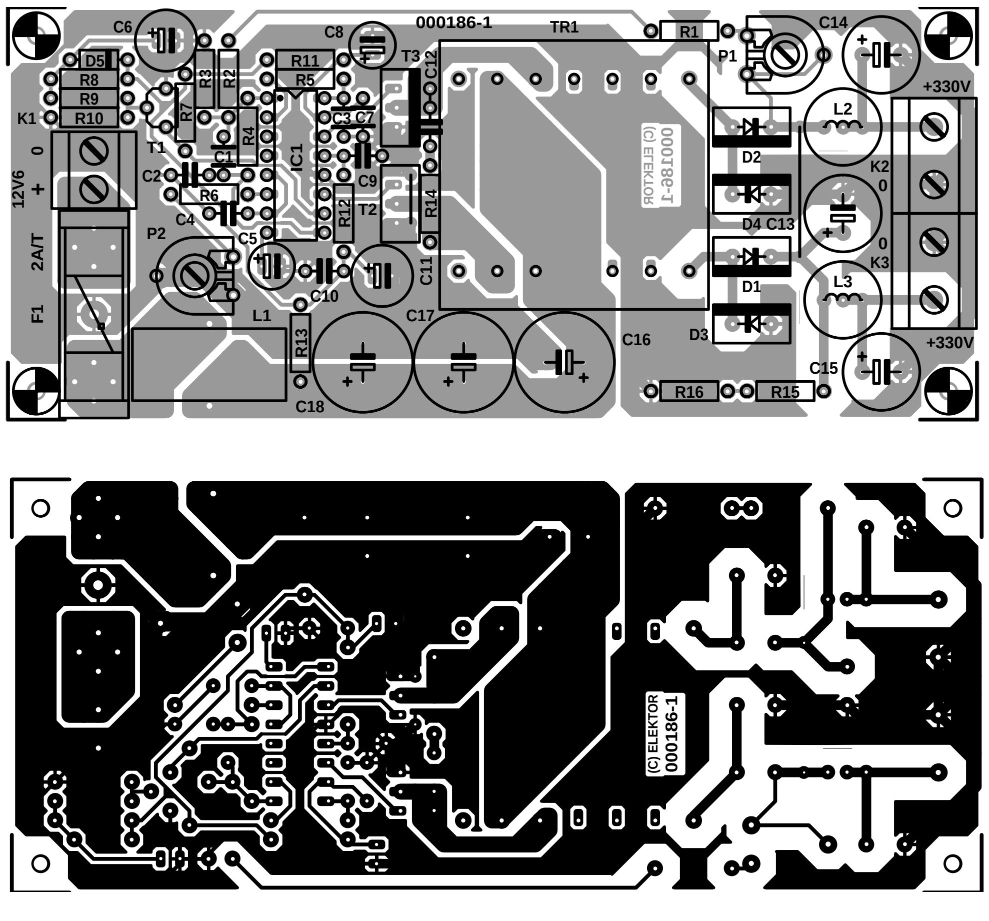


Figure 4. The PCB for the inverter is also tidy and compact.

We've used a standard ETD29 type former with N27 core material for the (home wound) transformer. The switching frequency has been kept relatively low (30 kHz) in order to save on smoothing capacitors at the primary side. Furthermore, three of them have been connected in parallel, which splits the current between them. This design can deliver a power of about 30 W.

The oscillator frequency can be set with P2 within a wide range (±7 kHz) to compensate for the tolerance of C4 (1 nF MKT), although the exact frequency isn't critical. To facilitate maximum power transfer, the dead time has been kept to a minimum by connecting the discharge output directly to CT and by keeping the value of C4 as small as possible.

The reference voltage is decoupled by C3 and fed to the non-inverting input of the error amplifier by R5. The output of the error amplifier (COMP) is also the input of the PWM comparator, which determines the pulse width. C2 limits the bandwidth and provides stability. The 330 V output voltage is fed to the inverting input of the error amplifier via potential divider P1/R1/R2/R3,

COMPONENTS LIST

330-V converter

Resistors:
R1,R2 = 120kΩ
R3 = 4kΩ7
R4 = 470kΩ
R5 = 1kΩ
R6 = 15kΩ
R7 = 6kΩ8
R8,R9 = 680kΩ
R10 = 330kΩ
R11,R12,R13 = 2Ω2
R14 = 100Ω
R15,R16 = 82kΩ
P1 = 100kΩ preset H
P2 = 10kΩ preset H

Capacitors:
C1 = 1nF ceramic, lead pitch 5mm
C2,C9 = 10nF ceramic, lead pitch 5mm
C3,C7,C10 = 100nF ceramic, lead pitch 5mm
C4 = 1nF MKT, lead pitch 5mm
C5,C6 = 100µF 16V radial
C8 = 100µF 25 V radial
C11 = 220µF 25V radial
C12 = 4nF7
C13,C14,C15 = 2µF2 450V radial, lead pitch 5mm, diameter 10mm
C16,C17,C18 = 1000µF 25V radial

Inductors:
L1 = suppressor coil 40µH 3A, type SFT10-30 (TDK)
L2,L3 = 47mH, e.g., 2200R series type 22R476 (Newport Components)

Semiconductors:
D1-D4 = BY329-1000 (Philips)
D5 = BAT85
T1 = BC550C
T2,T3 = BUZ11
IC1 = SG3525A(N) (ST Microelectronics)

Miscellaneous:
K1 = 2-way PCB terminal block, lead pitch 5mm
K2,K3 = 2-way PCB terminal block, lead pitch 7.5mm
F1 = fuse 2AT (time lag) with PCB mount holder
TR1 = ETD29 (Block) *
primary: 2 windings 11 x (3 x 0.5 mm parallel) ecw
secondary: 1 winding 300 x 0.3 mm ecw
PCB, order code **000186-1** (see Readers Services page)

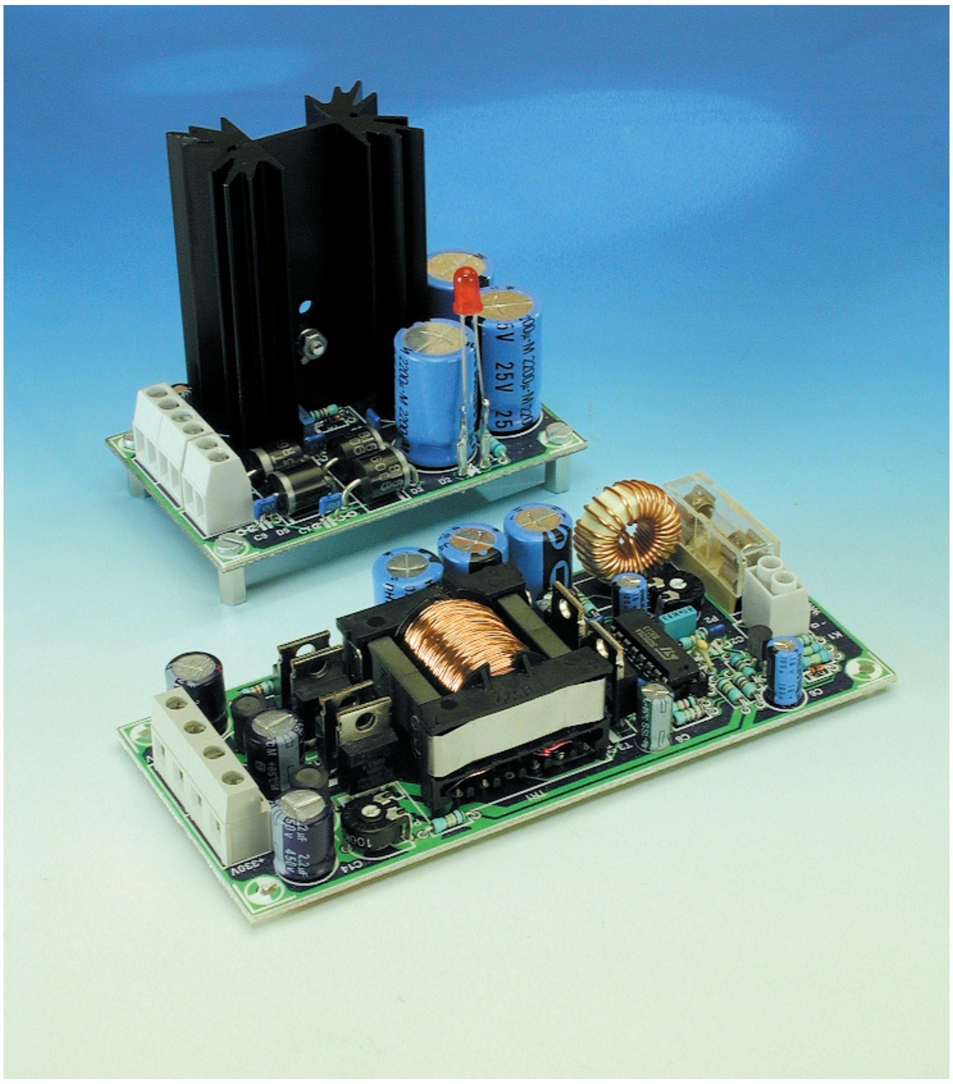
* see text

where R4 determines the open loop gain. P1 is used to adjust the value of the output voltage. The range has purposely been made fairly large (theoretically 270 V to 370 V), which gives the inverter plenty of scope for use in other applications. Slightly lower or higher output voltages can be obtained by varying the number of secondary turns proportionally (e.g. 273 turns would give 300 V). Keep in mind that if you use too many turns you can still get the correct output voltage, but at a reduced efficiency because the output is peak-rectified. The surplus energy will then be lost and dissipated in the transformer.

The modest circuit around T1 provides a delay of about 45 seconds between the application of the 12.6 V supply and taking the shutdown input low; this has to be below 0.6 V to enable the inverter. C6 is charged slowly by the potential divider of R8/R9/R10, which causes the voltage at the base of T1 to rise slowly and causes it to conduct. D5 causes C6 to discharge quickly when the supply is switched off.

R11, R12, R13 and C7-C11 are used to decouple the supply to the PWM regulator. R14/C12 reduce the spikes that are caused by the fast switching of transistors T2 and T3. The alternating voltage at the secondary is rectified by four fast soft-recovery diodes (D1-D4). Smoothing is carried out by 450 V radial electrolytics (C13, C14, C15). These are followed by low pass filters that reduce the ripple of the switching frequency even further. We've assumed that the inverter will be used with a stereo amplifier so we've provided two supply outputs, each with its own filter network (L2/C14, L3/C15). For the inductors we've used the 2200R-series from Newport Components, but the board will also accept the 8RB and 10RB series from Toko (available from Cirkit).

L1 filters the input supply and F1 protects the input supply from overload, which is important when, for example, a car battery is



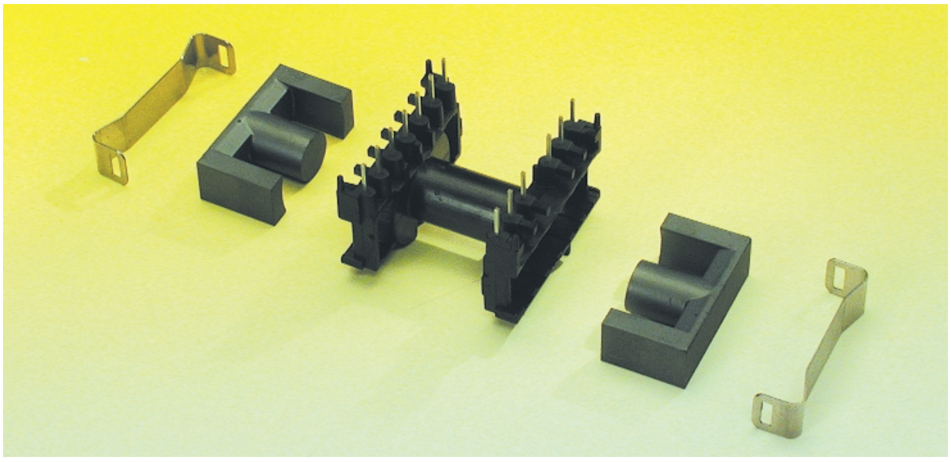


Figure 5. This shows an exploded view of the transformer parts.

used as source. For an even cleaner supply you should thread both 330 V cables through a large ferrite bead, which reduces common mode interference.

Construction of the inverter

The PCB for the 330 V inverter is shown in **Figure 4**. Because of the high voltages present, the layout is such that there is a minimum separation of 3 mm between the HV tracks and the earth plane. It is for this reason that the wire link between the cathodes of D1 and D2 is routed away from the low-voltage section (otherwise it could have been a track between the diodes).

Populating the board is simply a matter of carefully going through the parts list and soldering the components in place. The only part that could cause problems is transformer TR1. But this isn't as complicated as it might appear, since we can use a transformer kit supplied by Block (EB29 — see **Figure 5**). This also contains a pre-cut insulating foil, which is used to isolate the three windings from each other. The laying of subsequent windings is made easier by the tightly placed insulating foil. The ends of the secondary windings can be covered with the supplied insulating sleeve, which reduces the possibility of shorts between the windings.

The two primary windings consist of 11 turns of three strands of 0.5 mm enamelled copper wires, which are wound in parallel (next to each other) as if it was one conductor. The first winding is made between pins 3 and 11. This should cover the former with one layer of 0.5 mm copper wire. A layer of insulating foil is placed tightly across this, after which the next primary is wound between pins 2 and 12. This too is covered with a layer (or two) of insulating foil. Both primary windings have to be wound in the same direction to make sure that they are more or less identical. This ensures that the field has the smallest possible offset, which improves the performance of the transformer.

If you make the secondary winding very carefully, it is possible to use 0.4 mm wire, which reduces its resistance (resulting in better efficiency). But note that a sloppily wound 0.3 mm winding can fill the former almost completely. The transformer doesn't have an air gap!

And finally…

Once both boards have been populated and tested, they can be mounted with a 15 V transformer in an enclosure. It might appear easiest to mount the supply and Valved RIAA Preamplifier in one enclosure, but to obtain the best quality, and to keep interference to a minimum, we would recommend that the supply and the preamplifier are mounted in separate enclosures. If you do decide to mount them in one enclosure, you should at least have a metal screen between the supply and preamplifier sections, and the distance between the boards should be made as large as possible.

When we tested for interference suppression in our lab we weren't disappointed. When the Valved RIAA Preamplifier was used in conjunction with this power supply we measured the 60 kHz component at –90 dB, which is below the noise level of a typical phono signal. The 30 kHz component caused by the field of the inverter was at a level of –110 dB. Both measurements are relative to an output signal of 200 mV.

(000186)

Relay Driver

Operation and design

By Karel Walraven

In many circuits the switching action is performed by a relay, which in turn activates an external load. This part of the circuit is rarely covered in the accompanying text, but there is more than meets the eye if you want to design a relay driver properly.

We should really look at the relay driver as a type of interface. After all, it forms the link between the control circuitry or PC and the outside world. The relay is used by the control circuitry to switch external appliances, lamps, pumps, sirens, etc on and off.

Because they're used so often, the relay driver is as common as the integrated voltage regulator, an LED driver or a power-up network. The description usually skims over the operation of these basic circuits. We feel we should look at the relay driver circuit in more detail.

Just five components

Apart from the bypass capacitor, the circuit for the relay driver contains only four or five components, as shown in **Figure 1**. It's all very simple really, but all components need to be chosen according to their function. Next we'll look in detail at each of the components.

Relay
First we should determine what load we're switching; for example a 100 W lamp operating at 240 V AC. A suitable relay would be the E-Card relay (made by Siemens) which we've often used before. This can switch 500 VA at a maximum current of 4 A and a maximum voltage of 250 V AC.
But note that you can't just multiply the maximum current by the maximum voltage (4 A x 250 V = 1000 VA), since the manufacturer rates the relay at a lower power.

This relay is available for many operating voltages: 5, 6, 12, 24, 48 or 60 V. The 12 V relay is often used, since the required voltage can be obtained from an inexpensive mains adapter. It is also very simple to derive a secondary voltage of 5 V from the 12 V mains adapter should this be required. Deriving this from a 6 V adapter would be much trickier.

The manufacturer's datasheet tells us that the resistance of the coil of the 12 V relay is 330 Ω (alternatively, we could have found the resistance using a multimeter). When we use this 12 V relay, the current through it will be 12 V / 330 Ω = 37 mA.

Transistor
When the operating voltage of the relay is 12 V, the transistor should be able to cope with at least this voltage across the collector-emitter junction. To be on the safe side, we'll choose a value that is half as much

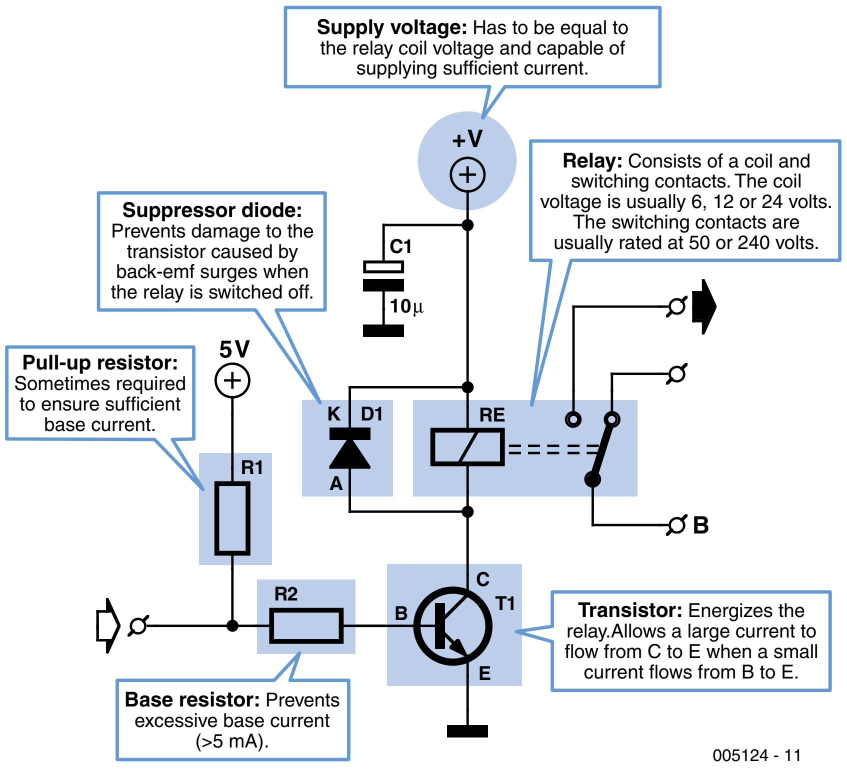

Supply voltage: Has to be equal to the relay coil voltage and capable of supplying sufficient current.
+V
C1
10µ
Relay: Consists of a coil and switching contacts. The coil voltage is usually 6, 12 or 24 volts. The switching contacts are usually rated at 50 or 240 volts.
Suppressor diode: Prevents damage to the transistor caused by back-emf surges when the relay is switched off.
Pull-up resistor: Sometimes required to ensure sufficient base current.
5V
K
D1
A
RE
R1
B
R2
C
T1
E
Transistor: Energizes the relay. Allows a large current to flow from C to E when a small current flows from B to E.
Base resistor: Prevents excessive base current (>5 mA).
005124 - 11

Figure 1. Circuit of the relay driver. The values chosen for the components depend on the size of the load to be switched.

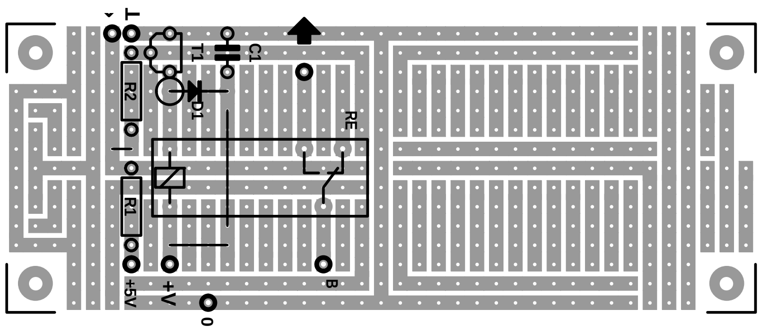


Figure 2. The relay driver is easily built on a UPBS-1 board.

Table 1

Transistor	VCE0 (V)	Ic (mA)	Hfe (times)
BC546	65	100	110-800
BC547A	45	100	110-220
BC547B	45	100	200-450
BC548C	30	100	400-800
BC517	30	500	30000
BD139	100	1500	40-250

Table 1. Some characteristics of suitable transistors

Table 2

Logic family	Sink (mA)	Source (mA)
TTL	16	0.4
LS	4	0.4
4000 (CMOS)	0.5	0.5
HC/HCT	4	4

Table 2. The source- and sink-currents of several well known logic families

again, which is at least 18 V.
The transistor also has to be capable of switching the required current; in our example that is 37 mA. In this case we'll use a safety margin of 100%, so choose a transistor which can switch at least 74 mA.

In this example it makes sense to use the smallest and most commonly used transistor: the BC547. This can switch 45 V and 100 mA.

Base resistor (R2)
Table 1 shows the most important characteristics of several popular transistors. The BC547 is made available in two versions; the A-type has a gain (H_{fe}) of 110-220 and the B-type 200-450. To be certain that the circuit will work under all conditions we'll use the lowest gain (110) when calculating the value of the base resistor (R2). The relay current goes into the collector and is 37 mA in our example. The gain of the transistor is 110, so we'll need a base current of 37 mA / 110 = 0.34 mA. As a rule of thumb we'll assume that the output that drives the transistor can deliver at least 3 V. The voltage across the base resistor is then 3 V minus the base-emitter voltage (0.65 V). The value for the resistor is therefore (3 V – 0.65 V) / 0.34 mA = 6910 Ω. We'll round this down to the nearest value resistor in the E12 series, which is 6k8.

Pull-up resistor (R1)
Table 2 shows the source and sink current capabilities of several popular logic families. When the source current is less than the required base current, the difference has to be supplied by a pull-up resistor (R1). As you can see in **Figure 1** it's not connected to the 12 V relay supply, but to the 5 V logic supply! The value of the pull-up resistor is calculated as follows:

Determine how much current is required for the logic family used. If we used LS logic the value for our example would be 0.34 mA – 0.4 mA = –0.06 mA. The negative result means that there is enough current, so we don't need a pull-up resistor in this case. Had we used a 6 V relay with an 80 Ω coil we would need a base current of 75 mA / 110 = 0.68 mA. The difference of 0.68 – 0.4 = 0.28 mA would then have to be provided by the pull-up resistor. From our earlier rule of thumb we know that the voltage across the pull-up resistor is 5 V – 3 V = 2 V. This gives a resistance of 2 V / 0.28 mA = 7140 Ω. Again we choose the nearest value in the E12 series, which is 6k8.

We just have to check that the driver IC can pull the base and pull-up resistor to ground. The current flow would then be 5 V / 6800 Ω = 0.735 mA. This value has to be less than the given sink current. In our example for the LS-family that is the case. But note that we couldn't have used a device from the 4000 series! In this case we would have to choose a transistor for T1 with a higher gain, as that would reduce the required base current.

Flyback diode (D1)
The function of the flyback (or 'suppressor') diode is to protect the transistor. The relay coil has the property (like all inductors) that it resists a change in the current flow through it. So when a voltage is applied across a coil it will take a while for the current to flow, but the reverse is also true: when the voltage across a coil is suddenly removed the coil will attempt to keep the current flowing at all costs. In practice the means that at the point when the transistor is switched off the coil becomes a generator and will supply a voltage with a reversed polarity. That causes a voltage to appear at the collector that is equal to the supply voltage plus the voltage that is induced across the coil. A risky business really, but the diode limits the induced voltage to 0.7 V, which prevents the transistor from being damaged.

Construction

Usually the relay driver will be part of another circuit and will not be built independently. But in the case when an existing circuit has to be extended with a switched output, it should be fairly easy to mount the five components of the relay driver onto a piece of Veroboard. We've figured out how to mount the circuit onto a Universal Prototyping Board size 1 (UPBS-1). As you can see from **Figure 2** it went very well and you'll be pleased to know that the (unstuffed) UPBS-1 may be obtained through our Readers Services.

Conclusion

These explanations and instructions make it fairly simple for anyone to design a relay driver module to their own specification. We would just like to point out that all calculations include a reasonable safety margin. The ICs can usually provide a higher current, so you don't have to worry that something will go up in smoke when the calculations have been a bit tight.

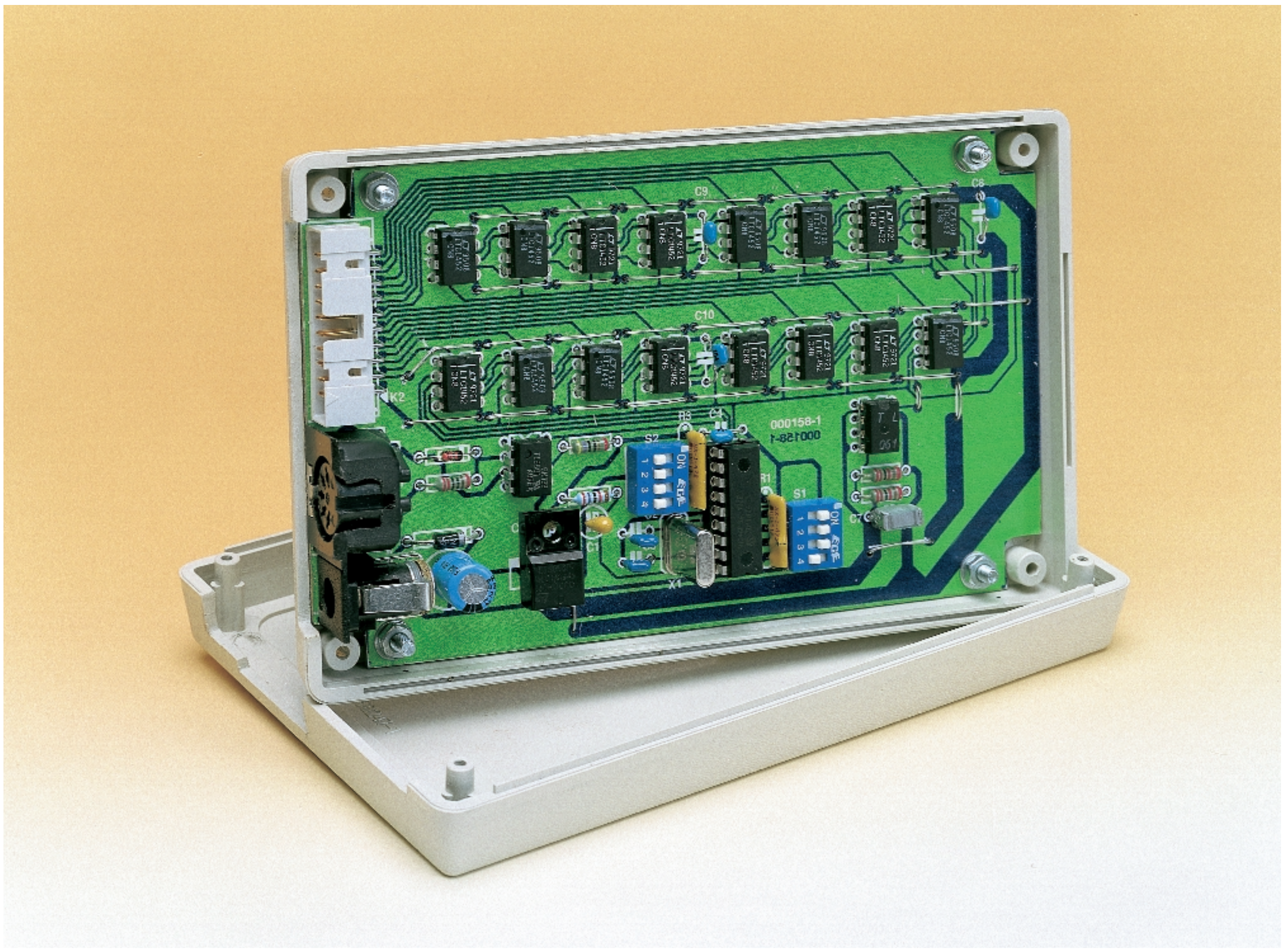
(005124-1)

MIDI to Voltage Converter

with 16 independent voltage sources

Design by J. Scherpenisse

Although MIDI (Musical Instrument Digital Interface) was originally developed for controlling musical instruments, it can also be used for numerous other purposes. In the circuit presented here, MIDI signals are used to set a number of DC voltages. Up to 16 output voltages can be independently adjusted.



Nowadays, every computer has a sound card, and the sound card always has a MIDI interface. However, in most cases the MIDI interface is not used. Most people control their musical instruments using MIDI commands (if they actually have any musical instruments…).
A microprocessor forms an excellent starting point for building a circuit that responds to MIDI commands. In this case, the author has developed a circuit that allows the values of several DC voltages to be set in a simple manner. These could be used for measurement applications, for example. From 1 to 16 channels are possible, with each channel having its own D/A converter and an adjustable output voltage ranging from 0 to 5 V.

One PIC with many converters

The circuit is built around a Microchip PIC16F84 microcontroller and 1 to 16 Linear Technology LTC1452 D/A converters (see **Figure 1**). There is thus a separate D/A converter for each output, instead of a single D/A converter multiplexed over several outputs. The major advantage of this approach is that it dispenses with the multiplexer and sample &

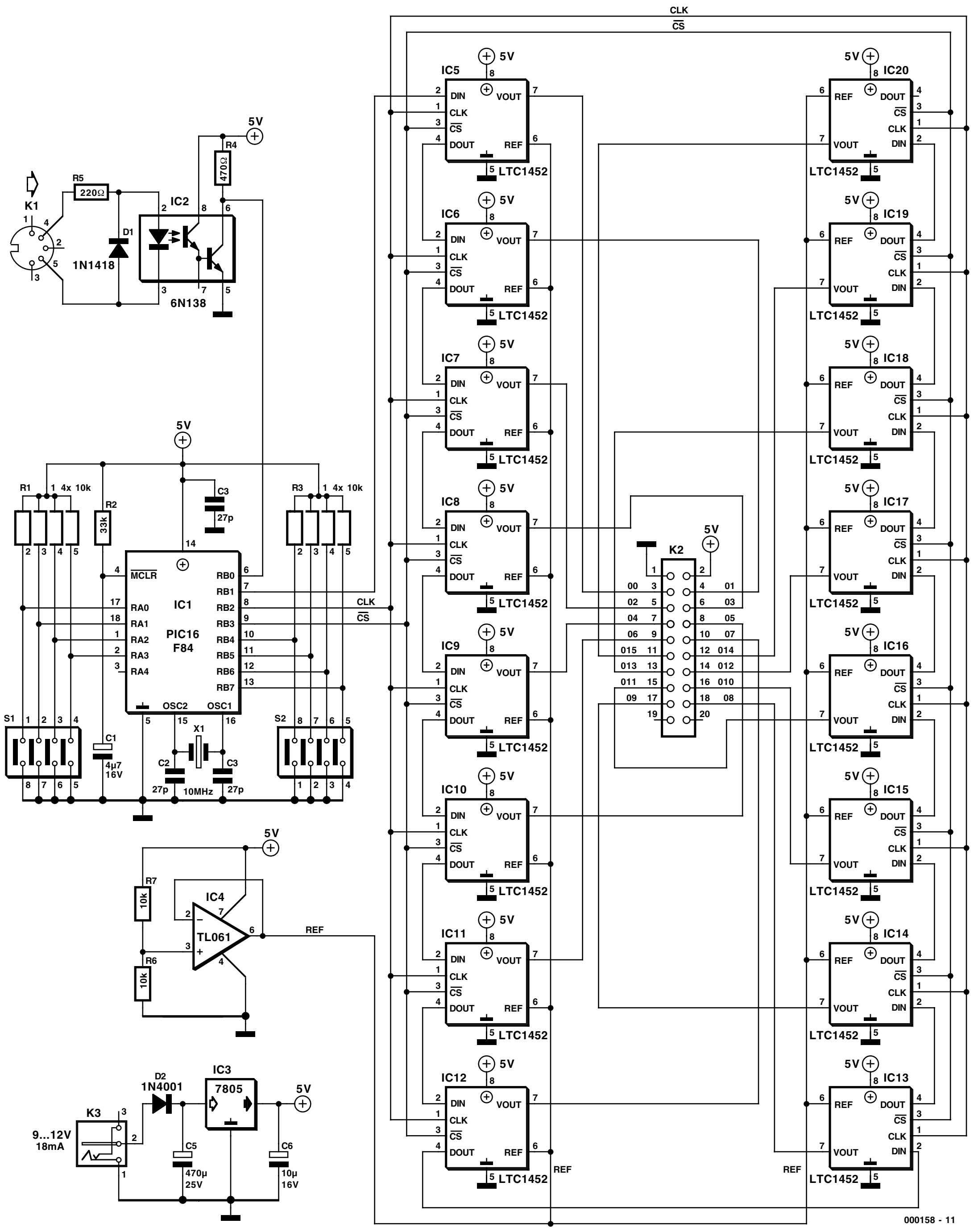


Figure 1. The core of the circuit consists of a PIC16F84 and up to 16 D/A converters.

hold circuits. The latter, in particular, are rather sensitive with regard to correct timing of the sample and hold signals.
The LTC1452 has a serial shift register at its input, which is followed by a latch register that is coupled to the actual converter. The shift register also has a serial output, so several converter ICs can simply be connected in series.
With a supply voltage of +5 V, the LTC1452 output can directly deliver a voltage between 0 and 5 V. For higher output voltages or heavy loads, a buffer must be added.
The interrupt routine in the processor scans the MIDI input signal to see if any 'control-change' messages with controller numbers between 48 and 64 (for example) appear on channel 1. If such a message is found, the 'value' parameter (the third byte of the message) is stored in a table. In the time between interrupts, the main program loop continuously sends the contents of this table, bit by bit, to the converter ICs via output port B1 of the processor. Following each data bit, a shift pulse is generated on port B2. After all data bits for the 16 converters have been output, a latch pulse is sent via B2. This transfers the data in all 16 converters to their internal latches. Since the data for the last (16th) converter are sent out first and the data for the first converter are sent out last, it is possible to omit one or more converters without making any changes to the remaining hardware or software.
The LTC1452 converter has a resolution of 12 bits, of which only 7 bits are used here. This results in 128 steps, which means that the step size is 39 mV when the maximum output voltage is 5 V. Standard MIDI controller values have seven data bits. However, it is possible to use 14-bit controllers with MIDI; this requires two controller addresses,

MIDI in brief

For true connoisseurs of music electronics, no further explanation of MIDI is necessary. However, a brief explanation of the structure and function of the control-change messages is in order for readers who are not fully familiar with MIDI.

MIDI (Musical Instrument Digital Interface) is a protocol that transfers data from a transmitter (MIDI out) to a receiver (MIDI in) via a serial connection. In principle, MIDI is a system for unidirectional data traffic. Due to their musical (synthesiser) origins, these signals were designed as messages that enable or disable a musical note, change a synthesiser setting, modify a volume level or apply a frequency shift to all notes of a synthesiser ('pitch blend').

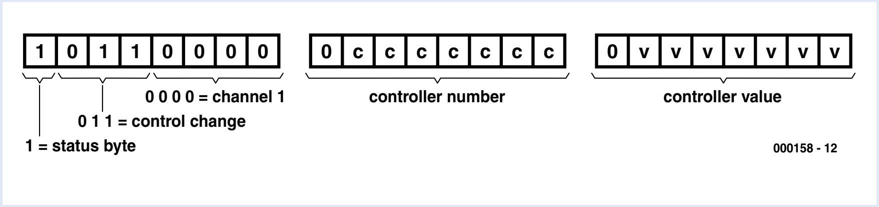
A MIDI message is usually made up of three 8-bit bytes. The first byte of each message (the status byte) indicates the message type and the MIDI channel for which the message is intended. There are 16 MIDI channels, so several synthesisers or several sounds within a single synthesiser can be separately controlled by a single computer using a single MIDI connection.

One group of messages is called 'control change'. These messages regulate controllers within a device (such as a synthesiser) for functions such as pitch blend, volume and portamento. When other types of devices (such as effects generators and reverb or delay units) are controlled via MIDI, control change messages can be used to modify delay times or reverb settings. A number of control change messages are predefined for a large number of synthesisers and accessory equipment.

The second byte of a control change message can address up to 127 controllers (the MSB of the second byte is always '0'). The third byte of the message contains the value that is sent to the controller in question. Here again the values range between 0 and 127, since the MSB of the third byte is also '0'.

Only the first byte of a message has a '1' in the MSB position. This means that the status byte can always be distinguished from the following bytes.

The control change messages used in this circuit thus appear as follows:



A MIDI message is always transmitted at a bit rate of 31.25 kbit/s, so one bit takes 32 μs. Each byte is preceded by a start bit (always '0') and followed by a stop bit (always '1'). The idle value between bytes is also '1'.

A 3-byte message thus contains $3 \times 10 = 30$ bits and thus takes 30×32 μs to send (approximately 1 ms in total).

In terms of hardware, MIDI connections are always current loops, with optocouplers used in the receiver to avoid ground loops.

COMPONENTS LIST

Resistors:
R1,R3 = SIL array 4 x 10kΩ
R2 = 33kΩ
R4 = 470Ω
R5 = 220Ω
R6,R7 = 10kΩ

Capacitors:
C1 = 4µF7 16V radial
C2,C3 = 27pF
C4,C7-C10 = 100nF
C5 = 470µF 25V radial
C6 = 10µF 16V radial

Semiconductors:
D1 = 1N4148
D2 = 1N4001
IC1 = programmed PIC16F84-10/p, order code **000158-41**, see Readers Services page
IC2 = 6N138
IC3 = 7805
IC4 = TL061
IC5-IC20 = LTC1452CN8

Miscellaneous:
K1 = 5-way DIN socket, PCB mount, receptacles at 180°
K2 = 20-way boxheader, angled pins
K3 = mains adapter socket, PCB mount
S1,S2 = 4-way DIP switch
X1 = 10MHz quartz crystal
Disk, contains source code and hex code files, order code **000158-11**, see Readers Services page

Program structure

The processor used in this circuit comes from the Microchip Inc. PIC series (http://www.microchipinc.com).

The PIC16F84 has 1 k words of 14-bit flash program memory, 68 bytes of data memory, 1 timer, 13 I/O ports and several interrupt options. The package has only 18 pins.

The software for the PIC16F84 is written in the CVSAM language from TechTools. This assembler language can actually be regarded as a collection of macros that translate single source code statements into one, two, three or sometimes even four true processor instructions. The new instructions that arise in this manner closely resemble standard instructions for 'big' microprocessors and are very easy to use. The actual instruction set of the PIC processors is also understood and correctly assembled by the CVSAM. In some cases, using the original instruction set is slightly more efficient. This can be a consideration with real-time applications and in case of interrupts that must be handled as quickly as possible. On account of the readability of programs written in CVSAM, the author has chosen this language in preference to the Microchip assembler (MPASM).

The optocoupler is connected to port B0, which is the standard interrupt pin. The start bit of each received byte thus triggers an interrupt. The interrupt routine polls port B0 two times in rapid succession, in order to eliminate any noise pulses that may be present. If the start bit is recognised, the port B0 interrupt is disabled and a new interrupt is enabled, namely that of the timer. The timer is then started, so that an interrupt occurs again after approximately 32 µs. In the subsequent routine, the value at the B0 input is stored and the timer is restarted. In this manner, the eight bits of the byte are read in. After this, the B0 interrupt is re-enabled, to allow the following byte to be received.

After each byte has been received, a test is made to see what type of byte it is. If it is status byte for a control change message, a switch is set to cause the following byte to be interpreted as a controller number. Following that byte, the switch is set to yet another value, so that the byte following the controller number byte will be seen as a controller value. Should a status byte be received at an incorrect time (when a data byte is expected), the bytes received up to that point are ignored and the cycle starts again from the beginning.

Once all data for a controller have been received, the table is updated with the current values.

As long as no interrupt routine is active, the main program loop looks after sending the data in the table to the D/A converters. The cycle time of the main loop is approximately 620 µs.

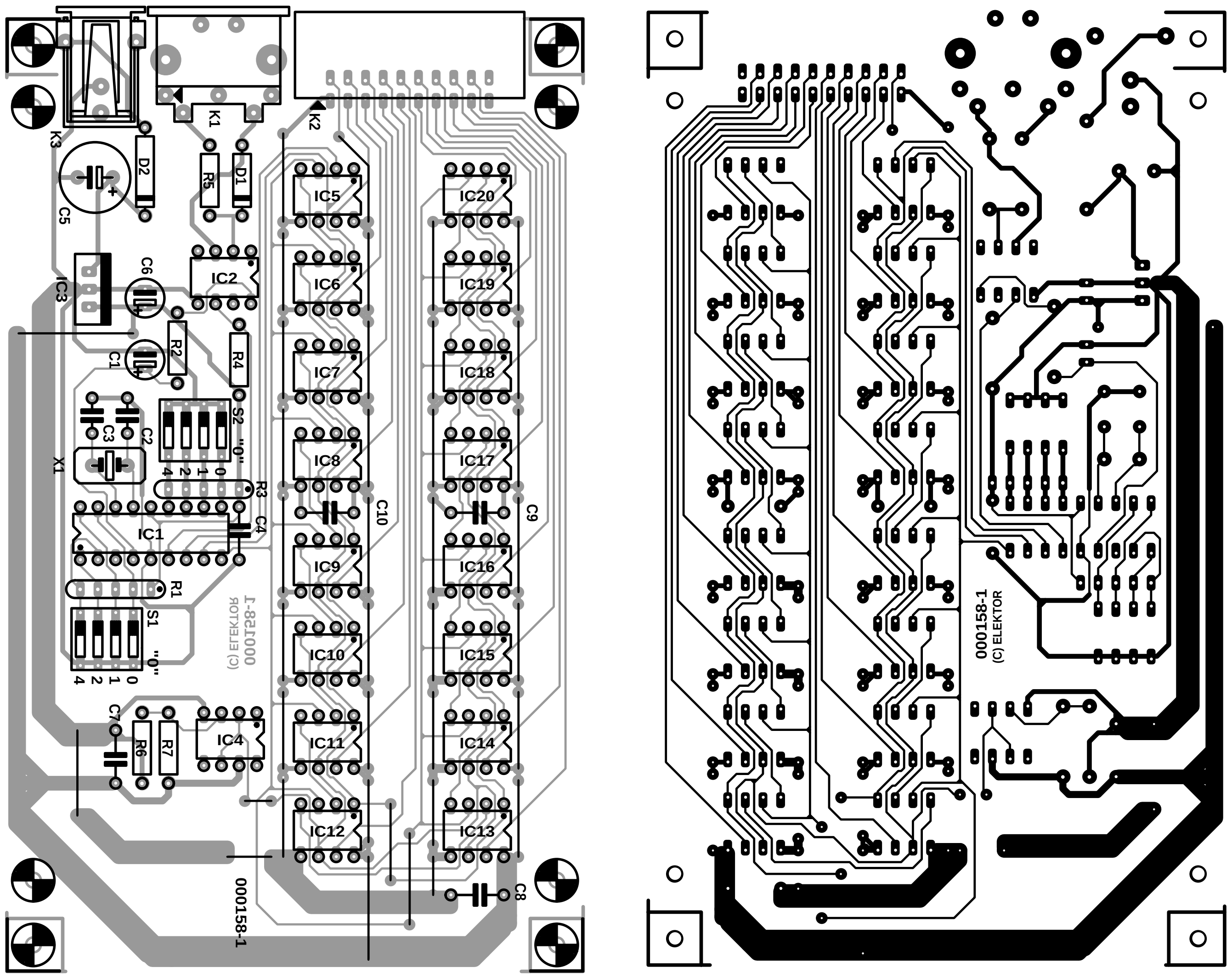


Figure 2. The single-sided printed circuit board for this circuit holds all components and connectors.

and thus two control-change messages. This capability is not (yet) implemented. Consequently, in this version only seven bits are sent for each DAC, and they must be padded with five null bits and five extra shift pulses per converter.

The reference voltages for the converters are set to a value of 2.5 V, using the voltage divider R7/R6 and the buffer IC4. This yields a maximum output voltage of 5 V. The output signals from all the converters are available on connector K1.

Four DIP switches are connected to port A of the processor. These can be used to set the controller numbers. The binary value of port A (four bits, with A0 = LSB) is shifted three bits to the left when the supply voltage is switched on. This yields steps of 8, thus 0, 8, 16, …, 120. This value is then used as the starting value for the controller numbers to which the circuit will respond. If all switches are closed, the circuit starts with controller numbers beginning at zero. If the switch for RA0 is open ('1'), the controller numbers begin with 8.

The MIDI channel number can be set using the DIP switches connected to ports RB4–RB7. This port is also read on power-up. If all four switches are closed ('0000'), MIDI channel 1 is selected. When all four switches are open ('1111'), MIDI channel 16 is selected.

The MIDI signal is fed to port RB0 of the PIC via DIN socket K1. Optocoupler IC2 provides electrical isolation between the MIDI source and the circuit, as is common with MIDI equipment.

A 10-MHz crystal with two capacitors is provided to generate the clock signal for the PIC.

There is also a 5-V regulator on the circuit board, so a simple unregulated mains adapter that provides 9–12 V (20 mA max.) can be used to power the circuit.

Construction

The printed circuit board layout shown in **Figure 2** has been developed for this circuit. The board is not available from Readers Services, but since the layout is single sided, it should not be too difficult for you to make the board yourself. When mounting the components, pay attention to the fairly large number of wire bridges, especially around the converter ICs. Mount as many converters as you need, starting with the first converter (IC5).

A programmed PIC for this circuit is available from Readers Services under order number 000158-41. If you want to program your own 16F84, you can download the source code file from the Elektor Electronics Internet site (*http://www.elektor-electronics.co.uk*), but it is also available on a floppy disk under order number 000158-11 for those few readers left with no access to the Internet.

Other than this, anyone with a bit of soldering experience should not have any trouble assembling the circuit. All connectors are located at one edge of the board, so it can easily be fitted into a suitable enclosure. A standard MIDI interconnect cable can be used to connect the circuit to the equipment that supplies the control data.

(000158-1)