THE ELECTRONICS & COMPUTER MAGAZINE

NOVEMBER 1999
£ 2·85

ELEKTOR ELECTRONICS

PC TOPICS:

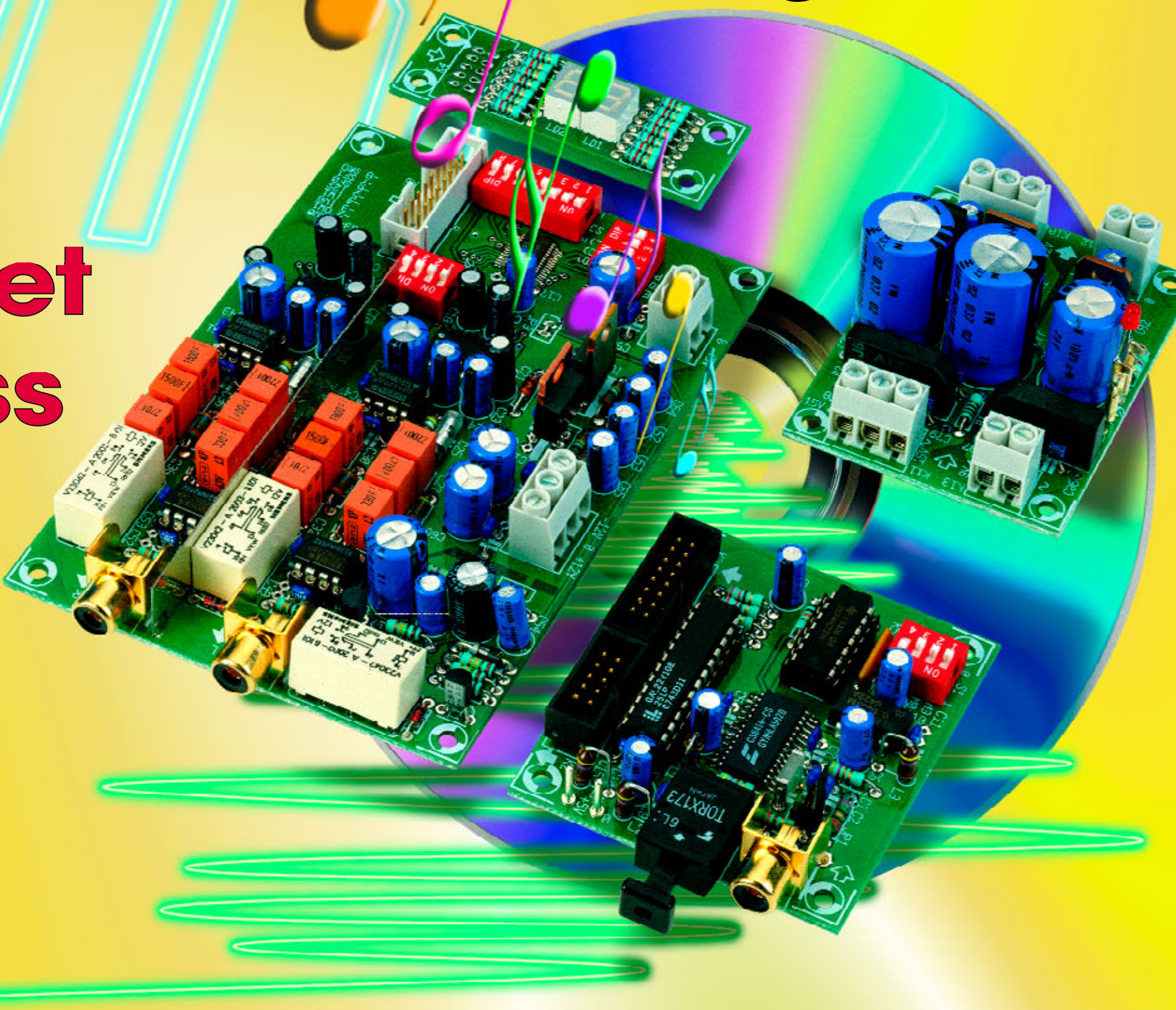
• recycling HD drive motors •
• micrologger with PC interface •
• PC connection cables •

TV line monitor

gradient meter

audio DAC 2000

96 kHz/24 bit sampling



fast Internet access by ADSL

November 1999 Number 282

Elektor Electronics
Volume 25
ISSN 0268/4519

CONTENTS

WHEN ELECTRONICS WAS YOUNG (9)

In 1895, the German physicist Wilhelm Conrad Röntgen (1845–1923) accidentally discovered a kind of radiation whose nature he could not determine, and which he therefore called X-rays. He experimented with burst ionization and used for this purpose a screen covered with a thin film of fluorescent material [BaPt(CN)4]. When X-rays were passed through a human onto a photographic plate, the bones were seen as shadowed areas against the lighter flesh, while metal objects, such as a ring, gave opaque shadows. Röntgen later suggested that X-rays were an electromagnetic radiation akin to light but of shorter wavelength, which was proved by von Laue in 1912. Röntgen was awarded the first Nobel Prize in Physics in 1901 for his discovery of X-rays. Their study added much to physics, gave a new technique for use in medicine, and, after the work of the British physicists Sir William Henry Bragg (1862–1942) and his son Sir William Lawrence Bragg (1890–1971) in 1915, led to X-ray crystallography. Sadly, Röntgen died in poverty during the period of high inflation in Germany.

Wilhelm Conrad Röntgen (1845...1923)

Edison found in 1884 that in an incandescent lamps in which a metal plate was placed opposite the filament a current flowed only if the plate was positive with respect to the filament. Five years later, Sir John Ambrose Fleming (1849–1945) showed that this current consisted of negative charges. In 1987, Sir Joseph John Thomson (1856–1940), while investigating cathode rays (the electrical discharge emitted from an electrode under high fields in a gas at low pressure), concluded that the rays were made up of particles, which were named shortly afterwards 'electrons' by the Irish physicist George Johnstone Stoney (1826–1911). Thomson also found that the charge-to-mass ratio (e/m) of these particles did not vary from one cathode material to another. The discovery of the electron was really the beginning of the 'age of electronics'.

Following on Thomson's work on cathode rays, the German physicist Karl Ferdinand Braun (1850–1918) modified a cathode ray tube so that its electron beam was deflected by a changing voltage. The resulting cathode-ray oscilloscope has been of inestimable value in electronics and other scientific work and is also the basic component of a television receiver.

As early as 1875, James Maxwell realized the potential possibilities of electromagnetic waves and he predicted wave propagation with a finite velocity, which he showed to be the velocity of light. His theories in the famous A Treatise on Electricity and Magnetism intrigued many researchers such as Heinrich Rudolf Hertz (1857–94), who in 1877 succeeded in producing electromagnetic waves experimentally, thus confirming Maxwell's predictions.

Guglielmo Marconi (1874–1937), an Italian physicist working in England, was interested in Hertz's experiments and began experimenting himself in 1895 with a spark induction coil and Branley coherer, and succeeded in sending telegraph messages over a short distance. He patented his invention in 1896 and in 1897 formed the Wireless Telegraph Company (later the Marconi Company) to exploit the patents and to set up the manufacture of spark transmitters and receivers.

Marconi continued to improve his equipment and succeeded in linking England with France by radio in 1899. On on 12 December 1901 he transmitted across the Atlantic from Poldhu in Cornwall to a kite-born antenna set up at St John's in New Foundland and so became a household word overnight at the age of 27. In 1909, Marconi shared the Nobel Prize for physics with Karl Ferdinand Braun. (995089)

Guglielmo Marconi (1874...1927)

stepper motor control

part 3: construction and software

Having built the PCB and secured it to the front panel, you should be ready to take the stepper motor control into use. This is best done in step-by-step fashion (pun intended), and we will guide you from adjusting the step-down voltage regulator right up to operating the PC terminal program. In the unfortunate case of the project not working as expected, help is available in the section 'Faultfinding'.

The step-down voltage converter is taken into use before fitting the integrated circuits and before securing the SMC board to 80C166 board. When an input supply voltage of about 10 V is applied to the converter, it should supply an output voltage of between 4.6 V and 5.4 V. Preset P5 is used to accurately set a level of 5.0-5.1 V. The converter's output voltage should remain stable when the input voltage is raised to 40 V. Two green LEDs, D11 and D12, light to indicate the presence of the supply voltage.

Once the step-down converter is know to function properly, the PCB is plugged on to 80C166 board fitted with the SMC firmware. Circuits IC13 and IC14 (74xx123) have to be present for indicating the clock signals and, later, for the stand-by mode. Additionally, a PC may be connected up to the serial interface. This PC should run a communication program set to ASCII transfer, 9600 baud, 8 databits, 1 stopbit, and no parity (9600,8,n,1).

After you switch on the supply voltage, the 80C166 board should produce its version and initialisation texts on the PC display. When the final *OK* appears, the stepper motor control software is ready for use.

Design by K.-.H. Domnick

Next, you have to verify that the SMC control software responds to pushbutton action, and supplies all clock and direction signals. On alternately pressing the *Left* and *Right* pushbuttons, the motor direction LEDs should respond accordingly by lighting or going out. The Tick (clock pulse) LED should light for the duration of the motor clock pulse.

Once this works as it should, the supply voltage may be removed and the GALs and power driver ICs installed in their sockets. Next, the stepper motors are connected up, one at a time, and each to its own power driver. Install the jumpers for the stepping order, current and standby mode, and then perform the analogue current adjustment with the potentiometers. Having done this the stepper motors should turn in both directions when the relevant control pushbuttons are pressed. If the motors respond properly to pushbutton control it should also be possible to control them using the PC.

If stepper motors are used with active zero-search, the motor direction has to be checked and modified if necessary. To change the direction, simply swap two wires of a phase winding.

Next, you may enable and test the sensor inputs by fitting the optocouplers. Whenever a voltage is applied to a sensor input (IN5-IN10, IC17 and IC18), the PC display should produce the associated report *s1* through *s6*. The same effect may also be achieved by short-circuiting the optocoupler outputs.

The state of 'Zero' (i.e., motor-home) inputs 1 through 4 (IC16) may be read in the same way as the sensor inputs. However, before this can be done, a zero (home) search command should be issued to the relevant motor. The PC display will then indicate the associated reports *n1* through *n4*.

FAULTFINDING

In all cases, check your soldering work, as this is the common cause of malfunctions. The same goes basically for the components mounted on the board. Further error sources include:

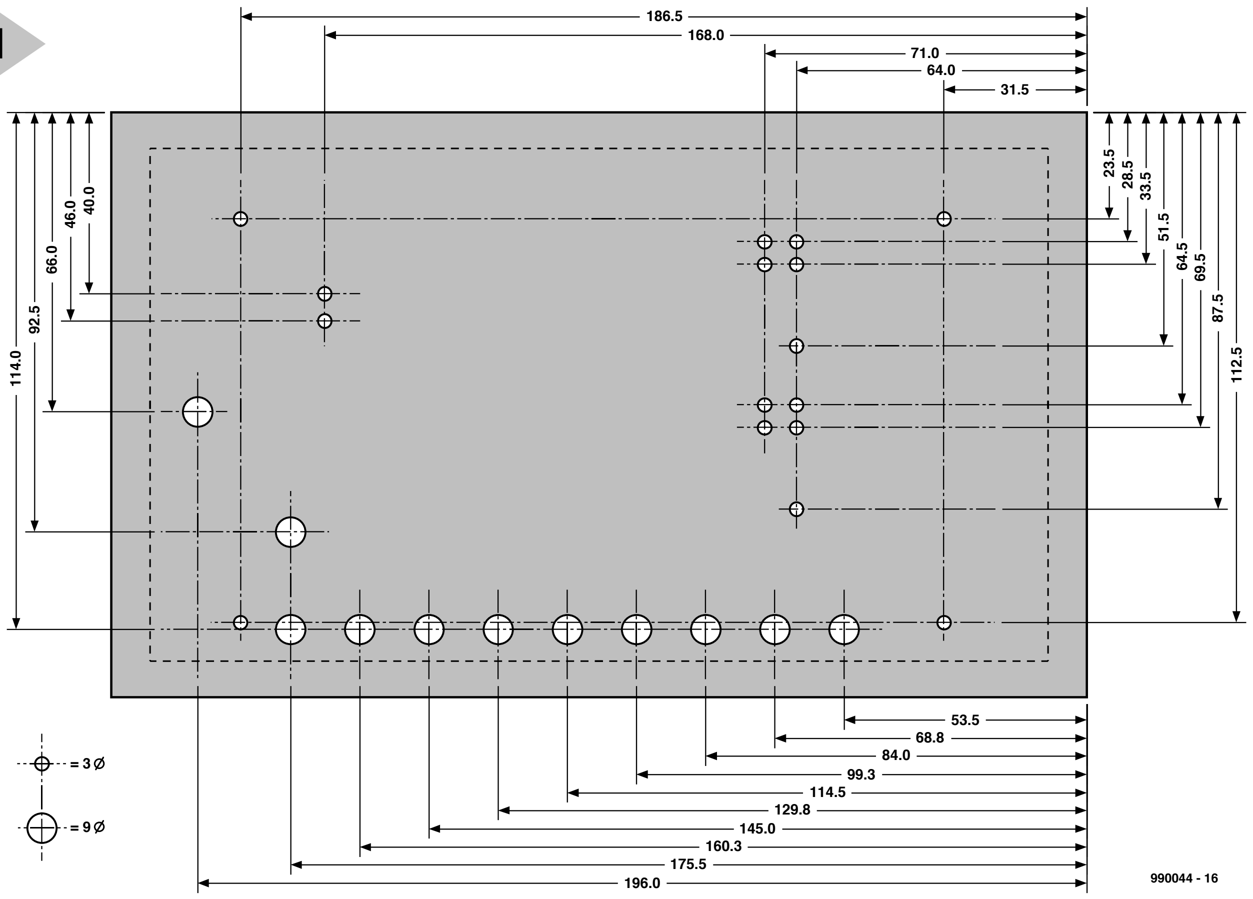
Motor does not operate at all

- Step-down voltage regulator does not supply 5 V.
- 80C166 controller board not connected.
- No SMC program in EPROMs, or H and L EPROM interchanged.
- Clock signal not available; lost between SMC board and 80C166 board (boxheader pin bent). Power driver ICs (IC1-IC8) missing.
- GAL (IC9-IC11) missing, not programmed or incorrectly programmed.

High-pitch sound heard, jerky movement of motor spindle, or no movement at all.

- Clock frequency or lower frequency too high.
- Too low current selected on jumpers JP9-JP20.
- With analogue current control, potentiometers P1-P4 set to wrong value, or jumpers JP9-JP20 not set

1



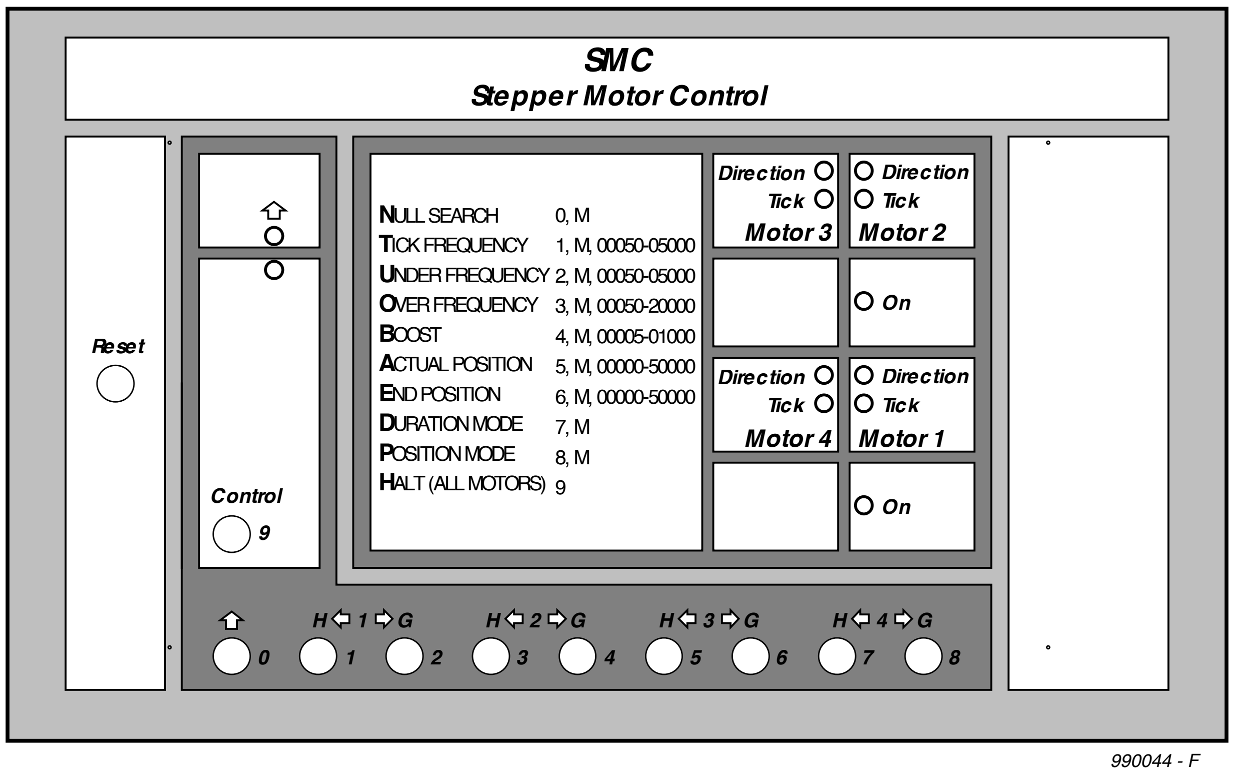


Figure 1. Front panel drilling details and suggested lettering.

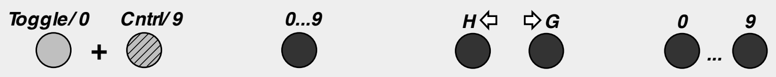
for full current.
➘ Too low supply voltage for stepper motor(s).

Motor runs briefly, then stops:
➘ Missing motor direction signal; missing or broken link with 80C166 board (boxheader pin bent).

Motor fails to find spindle home position:
➘ Wrong direction on stepper motor.
➘ Wrong polarity on home switch or sensor (normally-closed instead of normally-open contact).
➘ Missing optocoupler IC16.

Unable to communicate with PC:
➘ Wrong COM port selected on PC.
➘ Serial wires RxD and TxD interchanged.
➘ Jumpers for CTS and RTS missing, or wrong configuration.
➘ Missing SIO component or associ-

Table 1. Programming Mode



Function	**Pushbutton**	**Motor**	**Value = # sequence**	**Serial**
Tick Frequency	*1*	*M 1 - 4*	*00.050 - 05.000*	*T 50 - 5.000*
Under Frequency	*2*	*M 1 - 4*	*00.050 - 05.000*	*U 50 - 5.000*
Over Frequency	*3*	*M 1 - 4*	*00.050 - 20.000*	*O 50 - 20.000*
Boost	*4*	*M 1 - 4*	*00.005 - 01.000*	*B 5 - 1.000*
Actual Position	*5*	*M 1 - 4*	*00.000 - 50.000*	*A -2.147.483,648 - + 2.147.483.648*
End Position	*6*	*M 1 - 4*	*00.000 - 50.000*	*E -2.147.483.648 - + 2.147.483.648*
Duration Mode	*7*	*M 1 - 4*	-	*D 1 - 4*
Position-Mode	*8*	*M 1 - 4*	-	*P 1 - 4*
Halt (all motors)	*9*	-	-	*H 9*
Null search	*0*	*M 1 - 4*	-	*N 1 – 4*

Table 1. The pushbuttons on the SMC front panel have different functions in Programming Mode. Values to be entered into the SMC should include leading zeroes to ensure uniform length.

ated tantalum capacitors.

Nothing happens…
Further problems may be caused by construction errors on the SMC board or a missing, incorrectly specified or even defective part. Check your work thoroughly, or ask a 'second opinion'.

OPERATION

The ten pushbuttons on the SMC front panel allow direct and simple control of the stepper motors, as well as parameter entry. Complex control sequences are not possible using the pushbuttons — inevitably a personal computer is then called for.

Normal operation
SMC makes a distinction between Normal Operation and Programming Mode.

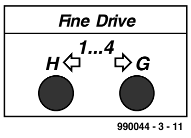


Figure 2. Normal operation without Toggle pushbutton pressed.

When pushbuttons 1…4 **Left** or 1…4 **Right** are pressed, the motor will turn in the selected direction at the selected clock frequency, until the relevant pushbutton is released.

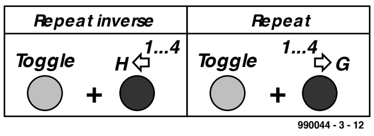


Figure 3. Normal operation with Toggle pushbutton pressed.

When the pushbutton 1…4 **Left** is pressed, the motor will turn in the opposite direction by the previous number of steps (equals command '*X — Repeat Inverse*). When the pushbutton 1…4 **Right** is pressed, the motor turns in the previously selected direction by the previous number of steps (equals command *Repeat*). The effect of Normal Operation with the Toggle switch pressed or not be interchanged with the aid of DIP switch number 2.

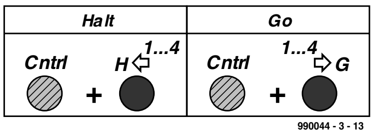


Figure 4. Normal operation with Cntrl pushbutton pressed.

Pushbutton 1…4 **Left** allows the stepper motor to be halted (equals command *Halt*). Pushbutton 1…4 **Right** starts the motor (equals command *Go*). Note however that a motor will only start if the values representing the current and end positions are different.

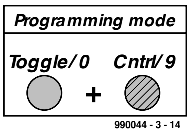


Figure 5. Enable/disable Programming Mode.

Programming Mode
To change from Normal Operation to Programming Mode, keep the Toggle pushbutton down while also pressing the Cntrl (Control) pushbutton. The yellow LED will start to flash.

To disable or terminate Programming Mode, repeat the above sequence. The yellow LED will go out.

Programming Mode is automatically ended after a complete or a wrong entry. With number entry, the Toggle pushbutton represents the value 0, and the Cntrl pushbutton the value 9. In Programming Mode, the pushbuttons have different functions.

When entering values into the SMC, leading zeroes should be included to make sure the values consist of the same number of digits.

SOFTWARE — INTERNALLY

All clock and direction signals are generated in software on the 80C166 controller board. Each clock is recorded as a 32-bit number and may be called up or modified as *Actual* (current) *Position*. After a successful 'home' search of the motor spindle, the relevant value is made 0. When the command *Go* is issued the controller computes a ramp (gradient) table from the parameters *Under Frequency* (lower frequency), *Over Frequency* (upper frequency) and *Boost* (acceleration rate).

Using these values, clock pulses are produced until the high-speed phase is reached. If the *Duration* mode (continuous operation) is enabled, all subsequent clock pulses take on the value assigned to *Over Frequency*. When the *Halt* command is issued, the values that make up the deceleration ramp (i.e., a look-up table) are processed until the motor stops.

If the spindle positioning mode is enabled, the processor also looks at the table entries representing the difference between *Actual Position* and *End Position*, as well as the number of steps

Table 2. Commands and reports

?	Help	-	Request help
A	* Actual Position	±2.147.483.648	Set/request current position
B	* Boost	5 – 1.000	Set/request motor acceleration (clocks per ms)
C	Copy Data	-	Copy data from memory into EEPROM
D	Duration mode	0/1 – 4/9	Switch on continuous mode (Position mode off)
E	* End position	±2.147.483.648	Set/request end position
E	Report	1 – 4	Report “End Position reached”
F	Fine drive	0/1 – 4/9	Turn slowly using clock frequency in selected direction
G	Go	0/1 – 4/9	Start motor / turn to end position
H	Halt	0/1 – 4/9	Halt motor
I	Info	-	Request info (status with semicolon preceding)
J	With zeropoint search	0/1 – 4/9	Turn to home sensor and then clear current position to 0
K	Without zeropoint search	0/1 – 4/9	No home sensor, clear current position to 0 immediately
L	Left rotate	0/1 – 4/9	Turn left using clock frequency (slow)
M	Motor actual *	1 - 4	Set/request motor no. for subsequent commands
N	Null search	0 / 1 - 4 / 9	Turn to spindle home (zeropoint) using clock frequency
n	Report	1 - 4	Report “Zeropoint reached”
O	* Over Frequency	50 - 20.000 Hz	Set/request upper frequency (fast)
P	Position Mode	0 / 1 - 4 / 9	Position mode on (Duration mode off)
q	Report	0 - 9	Acknowledge (q0) and error reports (q1-q9)
R	Right rotate	0 / 1 - 4 / 9	Turn right using clock frequency (slow)
S	Status	- / 0 / 1 - 4 / 9	Request status: M1, T500, U1000, O10000, B500, ...
s	Report	1 - 8	Report “Sensor reached”
T	* Tick frequency	50 - 5.000 Hz	Set/request clock frequency (slow)
U	* Under Frequency	50 - 5.000 Hz	Set/request lower frequency (Start/Stop)
V	Version.	-	Request program version number
v	Report	1 - 4	Report “Leaving zeropoint”
W	Repeat	0 / 1 - 4 / 9	Repeat previous no. of steps in previous direction
X	Repeat inverse	0 / 1 - 4 / 9	Repeat previous no. of steps in opposite direction
Z	* Int. position	± 2.147.483.648	Set/request intermediate position
z	Report	1 - 4	Report “Int. position reached”
	Possible values:	0	= current motor (set beforehand using M1 - M4)
		1 – 4	= indicated motor
		9	= all motors

for the high-speed phase. If more steps are required for the two ramps than for the difference between *Actual Position* and *End Position*, the motor will be unable to reach the high-speed phase.

Because all required clocks and steps have been computed after the start, all parameters are open to modifications (with the exception of the continuous or positioning modes).

If you want to stop a running motor before the spindle reaches the end position, you have to use the *Halt* command. It is possible to issue a *Null Search* command immediately after *Go*. On reaching the end position the motor will then automatically turn to the home position.

Another interesting command is *Int. Position* (intermediate position). When its associated value equals that of *Actual Position*, the program transmits a report (*z1-z4*) via the serial interface. This parameter may be modified at any time and as often as you want. Reports are also transmitted on reaching the end (target) position (*e1-e4*) or zero (null) position (*n1-n4*), on leaving the zero (null) position (*v1-v4*) or on detecting an active sensor input (*s1-s8*).

The command *Null Search* (turn to spindle home position) only causes the motor to actually turn to the home position if a ‘home’ sensor is available and the parameter *With zeropoint search* has been enabled. With *No home sensor*, only the current position is reset to zero, and a report (*n1-n4*) is transmitted.

PC COMMUNICATION

The SMC and the PC communicate through as serial link employing the following parameters:
9600 bits/second, 8 databits, 1 stopbit, no parity (9600,8,n,1). Using a terminal emulation program or general-purpose communication software for the PC (Telix, ProComm, HyperTerminal) you should be able to control and interrogate the Stepper Motor Control in plain ASCII. Complex sequences to be performed by the stepper motor will require a program, however, that supplies the relevant commands and evaluates the various reports returned by the SMC.

SMC wants to receive all commands either as a character or as a complete word without numbers or special characters. The system is not case-sensitive:

```
M1 / m2 / M3 /m4 / Motor 2
```

A line may contain more than one command provided a comma or colon (:) is used as a delimiter:

```
m1, t500 / M1, T500 /
M1:T500:M2:T800 / Motor 1,
Tickfrequency 500
```

After a semicolon (;) comment may be inserted up to the end of the line:

```
M1,T500 ; Clock frequency
500 Hz for Motor 2 <ret>
```

Status reports are transmitted in uppercase characters, while position reports appear in lowercase. Information and comment is always preceded by a semicolon (;). Each line ends with a Carriage Return and a Line Feed (CR-LF sequence).

All commands and reports are listed in **Table 2**.

(990044-2)

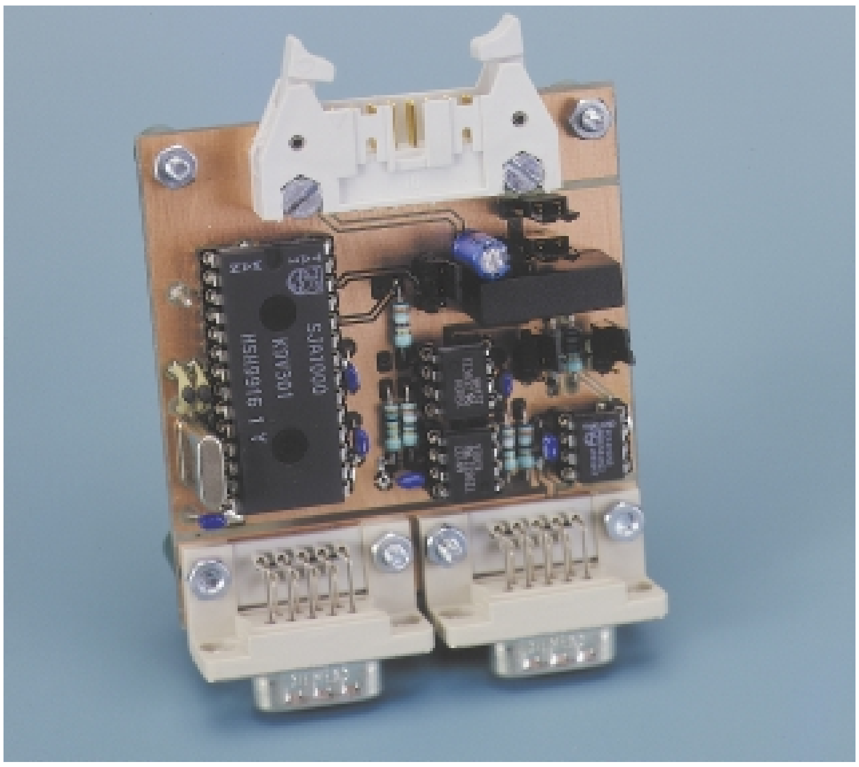
Table 3. Error reports

Q0	OK
Q1	Wrong command
Q2	Wrong motor number
Q3	Wrong value
Q6	RAM checksum error
Q7	EEPROM checksum error
Q8	EEPROM write error

controller area network (CAN)

intelligent, decentralized data communications in practice: Part 3

The first two parts of this article dealt with the history, standardization, basic setup, and data transmission protocol of the Controller Area Network. In this third part, the attention is shifted to more practical aspects. It deals with a network interface bus design that can be connected to any current microcontroller system.



By B vom Berg & P Groppe

INTRODUCTION

Current Controller Area Network (CAN) interfaces consist basically of three chips as shown in the block diagram in **Figure 9**. All the micro-controller is required to do is to write the data bytes (0–8) to be transmitted into the CAN protocol IC, fill the identifier field and the DLC field, and set the RTR bit accordingly. The remainder of the process:

- computing the CRC check sum;
- adding the remaining fields;
- accessing the bus;
- transmitting the data;
- detecting and remedying errors

is effected by the CAN controller IC.

The data are applied to the bus via the CAN transceiver IC that provides direct coupling to the bus.

The microcontroller then receives a message confirming the successful transmitting of the data or an error message, following which requisite action can be taken.

More or less the same happens when data.are being received. The CAN controller receives the CAN frames from the bus via the CAN transceiver IC, rechecks the check sum, removes all superfluous fields from the frame and passes either the received data or an error message to the micro-controller.

The reader will already have noticed that only a modest amount of material, hardware as well as software, is required for a CAN bus interface. Moreover, microcontrollers that contain a CAN controller on the same chip are already commercially available. Such a chip makes a two-stage CAN interface possible.

Some more information is given in the following section before the design of a practical interface can be discussed.

ACCEPTANCE FILTERING

We have seen in Part 2 that a Controller Area Network operating with standard frame format (CAN 20A) can process up to 2048 different identifiers. It is, of course, not necessary that each and every station linked to the bus can receive all data/remote frames. For instance, it may well be that for, say, Station K only frames with the identifiers 129, 1345, and 1999, are of interest and the other 2045 are of no consequence whatever. To avoid Station K receiving and processing all identifiers and passing them on to the microcontroller (which has to recheck whether each and every identifier is to be accepted or rejected – a time consuming activity), some kind of selection filtering of identifiers to ensure that only those of interest are passed to the microcontroller is highly desirable.

The selection of identifiers is called acceptance filtering. It allows the CAN controller chip to be programmed so that only frames with certain identifiers are passed to the microcontroller. All other frames are received and checked (incl. error correction), but not passed on. In this way, the microcontroller is freed of many superfluous comparisons and can therefore more speedily process the required data/remote frames.

There are two ICs that can be used for acceptance filtering.

BasicCAN

This IC has a simple filter that is, say, eight bits wide, which allows coarse pre-selection only. Normally, this consists of passing groups of identifiers, in, say, the range 700–707. Selection of a single identifier is not possible. This IC therefore requires the microcontroller to carry out a further selection to arrive at the wanted identifier.

9

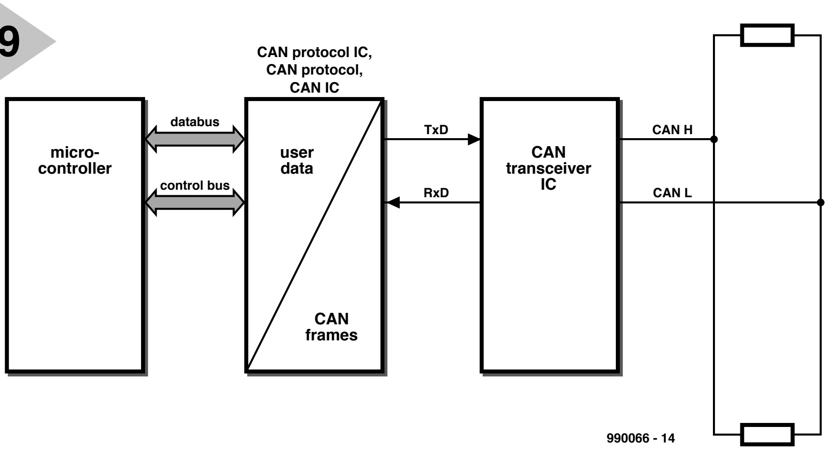


Figure 9. Block diagram of a three-stage Controller Area Network design.

Remote frames intended for the relevant station are also passed by the filter and applied to the microcontroller. It is only then that the microcontroller can generate the relevant response data and pass these to the CAN controller.

FullCAN

The FullCAN chip allows the exact programming and selecting of a single identifier. In other words, it can be set to accept a single frame or a number of single frames, for instance, only those with identifier 798.

The drawback of the IC is, however, that a large number of frames (with different identifiers) are not passed on since the controller programme is fixed.

Therefore, if many frames with different identifiers are to be received it makes better sense to use a BasicCAN chip. It must be borne in mind, however, that in this case the microcontroller needs to carry out a substantial part of the selection process and this in turn means that a more powerful microcontroller is needed.

A beneficial property of the FullCAN chip is that the microcontroller can program the response to a remote frame on to the CAN controller IC. When this IC receives a permissible

Table 4: Brief parameters of interface

CAN controller IC	*Type SJA1000 (Philips Semiconductors)*
Microcontroller interface	*can be arranged for Intel or compatible and Motorola or compatible microcontrollers*
Mode of operation 1:	*pin-, hardware-, and software-compatible with PCA82C200* *CAN 20A and 20B passive* *Standard frame format* *Data transfer rates up to 1 Mbps* *Acceptance filter BasicCAN*
Mode of operation 2:	*Extended and standard frame format* *Data transfer rates up to 1 Mbps* *CAN 20B capability* *Extended acceptance filter with BasicCAN properties*
Transceiver IC	*Compatible with ISO/DIS11898, high-speed CAN* *Data transfer rates up to 1 Mbps* *Internal protection against noise specific to motor vehicles* *Internal protection against short-circuits and thermal overload* *Non-powered nodes (stations) do not affect the bus* *Allows the design of Controller Area Networks with up to 110 nodes*

10

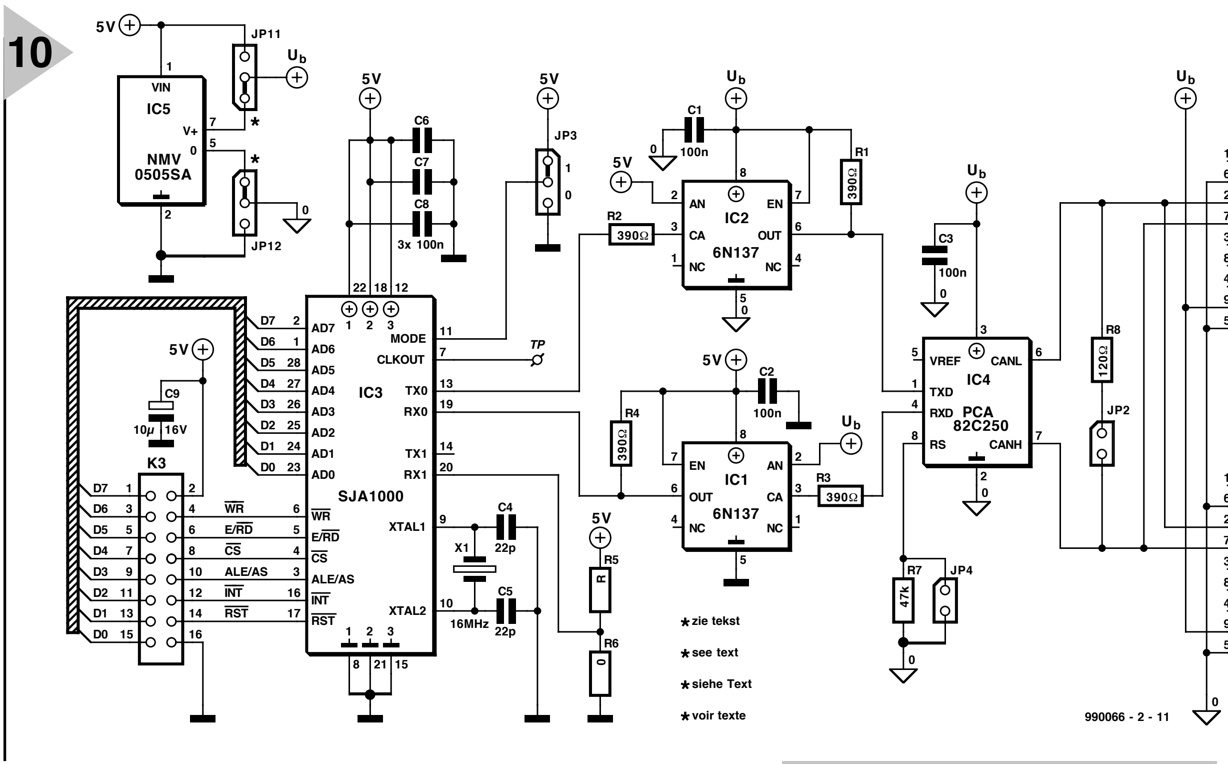


Figure 10. Circuit diagram of the CAN bus interface.

remote frame for the relevant station, it can send the response data frame without intervention by the microcontroller.

With the inexorable progress of the technology, the differences between the BasicCAN and FullCAN chips are becoming less defined. Also, FullCAN chips are becoming more powerful, so that they can select larger numbers of individual identifiers and can store more data registers. State-of-the-art CAN controller chips can switch between the two modes of operation by means of software.

COMPATIBILITY BETWEEN 20A AND 20B

As we have seen during the discussion of the frame formats (Part 2), there is a Standard Format with 11-bit identifiers and an Extended Format with 29-bit identifiers. Great care must be taken in choosing a CAN controller when both formats are used in a bus system (which is perfectly possible and permissible).

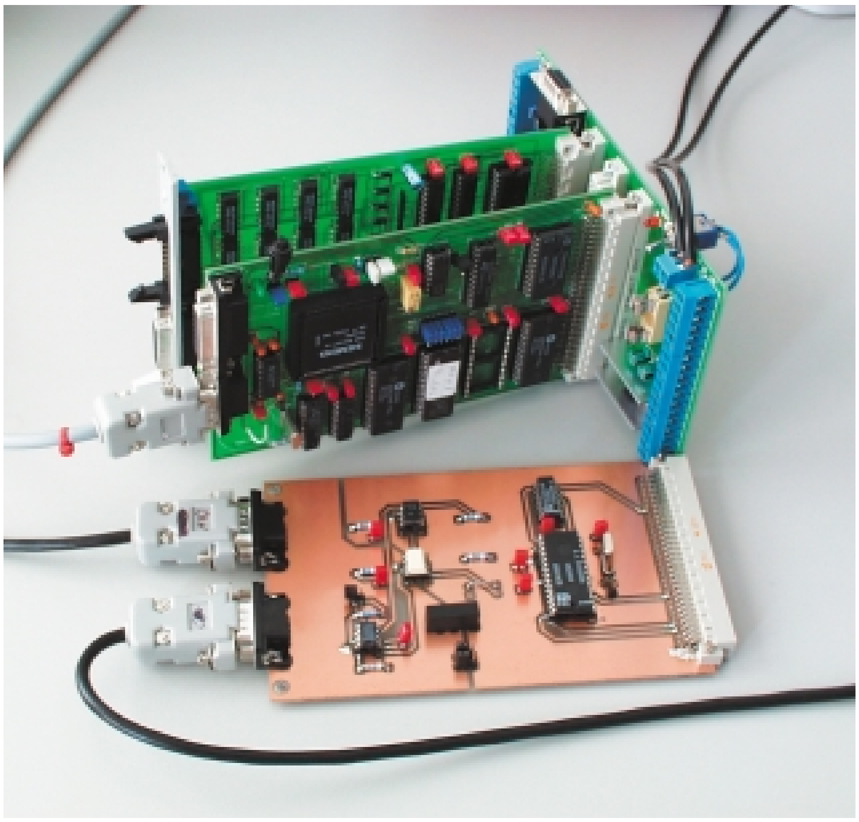
Controllers with 20A capability
These controller chips can process standard frames only and generate an error message when an extended frame message is received. Since this could bring the whole system to a standstill, these controllers can be used only in systems that operate with standard frames.

Controllers with 20A capability and 20B passive properties
These chips accept extended frames with 29-bit identifiers, carry out an error test, and respond with and ACK bit or an error frame.

Although the communication is not disturbed, the received extended-frame data are not stored or passed on, since the chips are intended for processing frames in standard format only. Nevertheless, these controllers are perfectly all right for use in hybrid systems.

Controllers with 20B capability
These controllers process, store, and pass on, standard-format as well as extended-format frames.

When a decision has to be made on the purchase of a CAN controller or microcontroller with on-chip CAN controller, there is such a wide choice



that it really pays to visit the Internet Home pages of producers like Hitachi, Intel, Motorola, NSC, Philips, SGS, Siemens, and Temic, and Texas Instruments to name but a few.

CAN BUS INTERFACE

After the lengthy discussions on theory and basic principles, the practical design of a CAN bus interface can now be described in relatively few words.

The circuit diagram of the interface is shown in **Figure 9** and a suitable printed-circuit board in **Figure 10**. Brief parameters of the interface are given in **Table 4**.

The CAN controller, IC_3, is a Type SJA1000, whose internal block schematic is shown in **Figure 11**. This chip is the successor of the PCA82C200 with which, in operating mode 1, it is pin-, hardware-, and software-compatible.

The interface with the microcontroller can be arranged for use with either a Motorola or an Intel chip (or compatible types).

The CAN transceiver, IC_4, is a Type PCA82C250.

The microcontroller is linked to the CAN bus interface by a length of flatcable, which should be not longer than 10 cm (4 in), terminated into 16-pin header K_3. The pin connections of this connector are given in **Table 5**.

Via this link the microcontroller exchanges operating data, control data, and status data, with the CAN controller. These data are processed by the controller both in the send and receive directions. The microcontroller therefore 'sees' the CAN controller as an extension to its memory to which it writes operating data to be transmitted, or from which it extracts received operating data.

The clock frequency, which is divided in several stages, can be measured at test pin TP, for example, when it is to be ascertained whether the controller can be accessed and programmed safely.

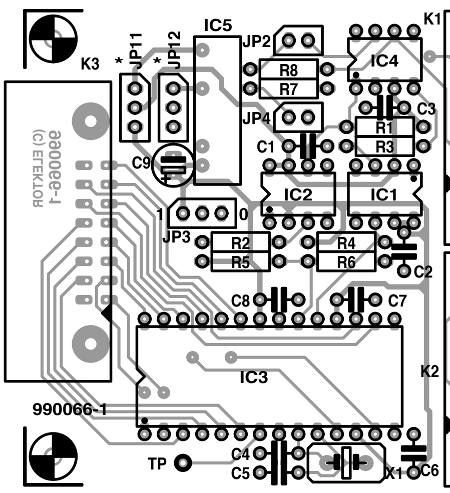
The serial output signal at pin 13 of the controller is applied to pin 1 of the transceiver via optoisolator IC_2. The transceiver generates the standard CAN bus signals which are available at its pins 6 and 7. These signals are impressed upon unshielded twisted-pair (UTP) copper wires via connectors K_1 and K_2.

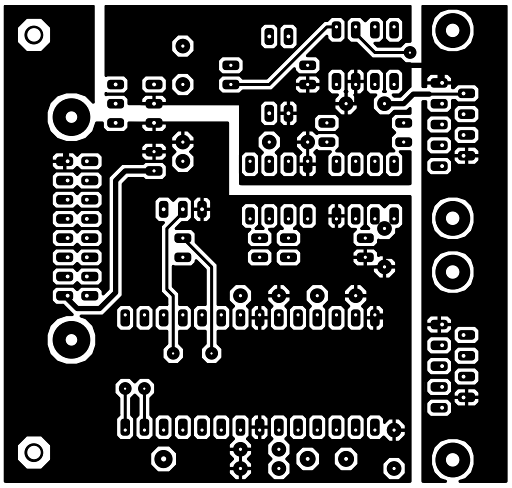
The signal received from the bus arrives at pin 4 of the transceiver from where it is applied to pin 19 of the controller via optoisolator IC_1. The controller converts the received bits and processes them in accordance with the relevant CAN protocol. The received signal is finally fed to the microcontroller where it is analysed.

Optoisolators IC_1 and IC_2, and 5 V DC/DC inverter IC_5, effectively isolate the microcontroller and bus sections of the node (station). The arrange-ment ensures that any faulty or uncertain signals on the UTP wires, although applied to the transceiver, cannot damage the microcontroller section and following system.

It is, of course, possible to build the interface without the isolating stages, so that R_1–R_4, C_1–C_2, IC_1–IC_2, IC_5, and JP_1–JP_2 can be omitted. The supply line terminals as well as the send and receive pins on IC_3 and IC_4 must then be interlinked as appropriate. It must be borne in mind, of course, that faulty

11





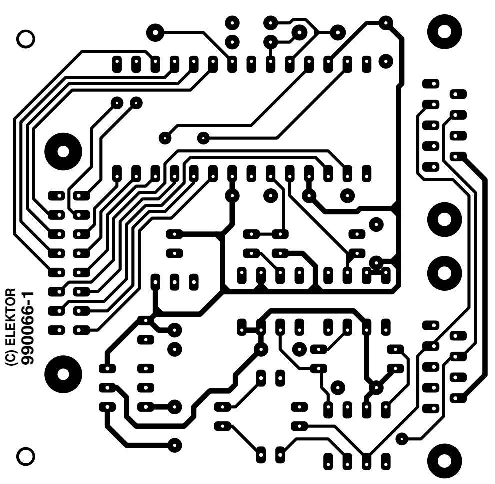


Figure 11. Printed-circuit board for the CAN bus interface.

Parts list

Resistors:
R_1–R_4 = 390 Ω
R_5, R_6 = see text
R_7 = 47 kΩ
R_8 = 120 Ω

Capacitors:
C_1–C_3, C_6–C_8 = 0.1 µF, ceramic
C_4, C_5 = 22 pF, ceramic
C_9 = 10 µF, 16 V, radial

Integrated circuits:
IC_1, IC_2 = 6N137
IC_3 = SJA1000
IC_4 = PCA82C250
IC_5 = NMV505SA (Newport/Farnell)

Miscellaneous:
X_1 = 16 MHz quartz crystal
K_1, K_2 = 9-way D connector, right-angled, for board mounting
K_3 = 16-way header, right-angled, for board mounting, with interlock
JP_2, JP_4 = 2-way, 2.54 mm pin strip and pin jumper (Maplin)
JP_3, JP_{11}, JP_{12} = 3-way, 2.54 mm pin strip and pin jumper (Maplin)
PCB Order no 990066-1 (see Readers' Services section toward the end of this issue)

Table 5: Pin connections of header K_3

Pin	Designation	Function
1	D7	
3	D6	
5	D5	
7	D4	
9	D3	
11	D2	
13	D1	
15	D0	
2	+ 5 V	positive supply line
4	WR\	Write\ signal
6	RD\	Read\ signal
8	CS\	Chip-Select\ signal
10	ALE	Address latch enable signal
12	INT\	Interrupt\ signal
14	RST\	Reset\ signal
16	GND	Earth connection

signals on the UTP lines are then applied unimpeded to the microcontroller section.

Jumpers JP_{11}, JP_{12}, and JP_2–JP_4 must be arranged as follows.

JP_{11}, JP_{12}
These jumpers (marked ∗ on the PCB) determine how the supply voltage is applied to the interface and the microcontroller system. With these jumpers set as shown in Figure 10, electrical isolation is provided and the supply voltage to the bus section (IC_1, IC_2, IC_4) is via IC_5.

In the other position of the jumpers, there is no electrical isolation, and all stages are supplied directly via the 0 and + U_b terminals.

An alternative in the latter case is the provision of supply voltage via two lines in parallel with the UTP wires and connecting these to pins 6 and 9 of K_2 and K_1 respectively.

If the jumpers are left open, there is electrical isolation, and the bus section is then powered via pins 6 and 9 of K_2 and K_1 respectively.

JP_2
When JP_2 is shorted, bus terminating resistor R_8 is connected between pins 6 and 7 of the transceiver. It must be borne in mind that only two bus terminators must be used, one at the beginning and one at the end of the UTP wires. Since more terminating resistors (at other nodes) will be in parallel with these two, they lead to a reduction in total terminating resistance, which results in a higher output current from the transceiver and this in turn can lead to thermal overload and damage or destruction of the chip.

JP_3
This jumper must be set according to which microcontroller interface is used. The position shown in Figure 10 (pin 11 of IC_3 at + 5 V) and marked '0' on the PCB allows Intel processors/controllers or compatible chips to be used.

When the jumper is placed in the other position (pin 11 of IC_3 at 0 V), Motorola processors/controllers or compatible chips may be used.

JP_4
This jumper, or rather resistor R_7, determines the slope of the edges of the pulses at the CAN bus.

In the case of high data transfer rates (up to 1 Mbps), steep-sloped edges are essential, but these lead to the risk of a large noise spectrum being generated by the CAN pulses. This noise can only be negated by the use of shielded twisted pair (STP) wires. The jumper must be used, thereby shorting out resistor R_7.

With low data transfer rates (up to 125 Kbps) the edges of the pulses need not be steep. This results in a narrow noise spectrum being caused by the CAN pulses, so that UTP (unshielded twisted pair) wires can be used. Jumper JP_4 must not be placed.

12

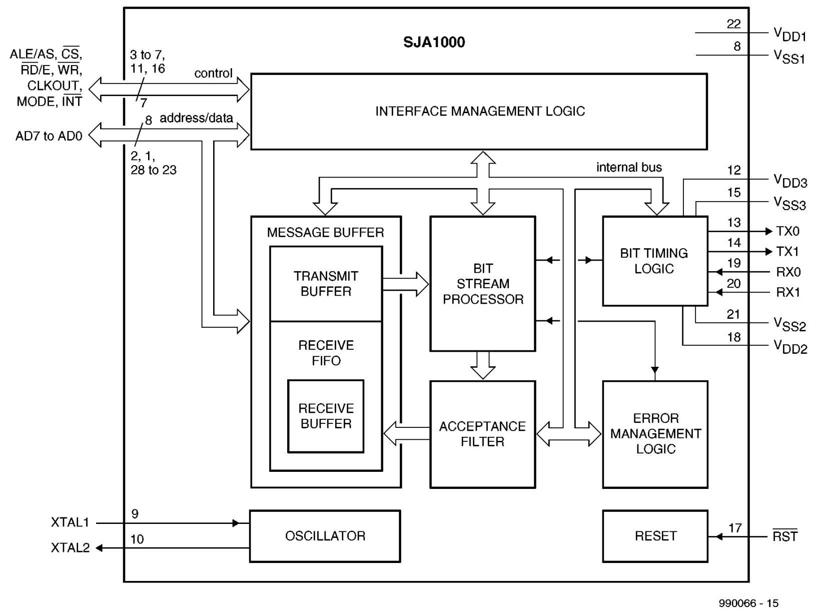


Figure 12. Block diagram of the internal arrangement of the Type SJA1000 Controller Area Network IC.

This concludes the description of the hardware for the CAN bus interface. A forthcoming article will deal with the connection of the CAN bus to a microcontroller system and its application in an experimental CAN bus system.

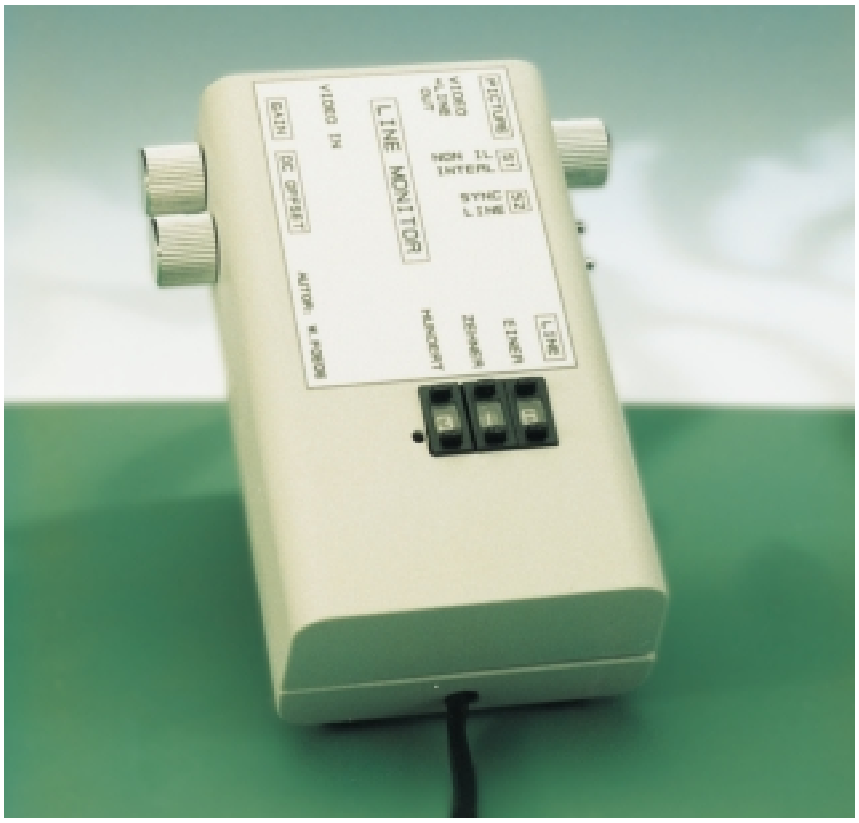
[990066]

enhanced TV line monitor

select and check video line contents

The tester presented here can do more than just generate an oscilloscope trigger pulse for a selected video raster line. It also marks the selected line in the CVBS signal and displays it on the TV screen as an oscillogram.

Design by W. Foede



Features:

- ✓ exact selection of one horizontal line (1 through 625)
- ✓ oscilloscope trigger output
- ✓ selected line marked in the passed-through video signal
- ✓ oscillogram of the selected line merged with the television picture; amplitude and position adjustable
- ✓ adjustable blend of original picture and oscillogram
- ✓ line content displayed either aligned to picture or offset with a horizontal sync pulse
- ✓ interlaced and non-interlaced operating modes

In order to be able to display a given line of a video image on an oscilloscope, you need a special trigger generator that uses the vertical synchronization pulse to detect the start of the frame or field (half frame) and then counts horizontal sync pulses until the desired line is reached, whereupon it generates a trigger pulse for the oscilloscope.

A circuit which does this (and only this) was published in the October 1994 issue of *Elektor Electronics* ('TV Line Monitor'). In that circuit, a microcontroller looks after all the counting and control tasks — everything except for sync separation. However, at the cost of only a little more hardware, this basic

1

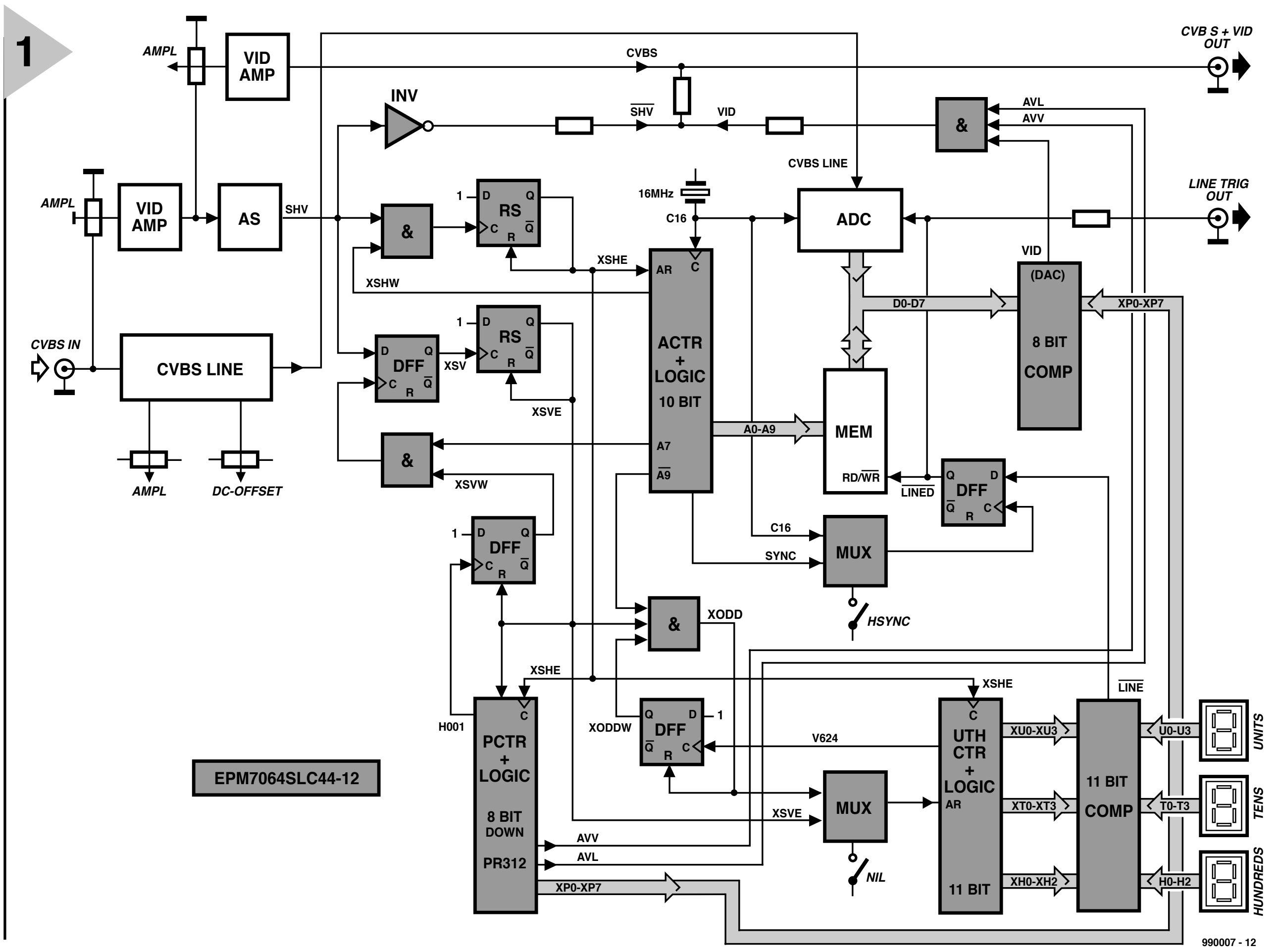


Figure 1. Block diagram of the enhanced TV line monitor. The shaded elements are contained in a ready-programmed EPLD IC.

function can be enhanced with a number of interesting extras (see the 'Features' box). Since this tester evaluates not only the sync pulses but also the content of the selected line, a different approach must be taken.

The video signal follows two different paths from the input. The first path leads to a video amplifier built using discrete components, in which the sync pulses are separated from the video signal so that they can be used to trigger the line counter. The second path is digital. First the CVBS (composite video) signal is sampled by an A/D converter, and the resulting data are stored in a fast RAM. A D/A converter restores the selected line to analogue form and merges it with the amplified CVBS signal in an output buffer amplifier. The enhanced TV line monitor also offers a range of operational settings, and it produces the external trigger signal that is essential for making measurements with an oscilloscope.

Analogue and programmable logic

The block diagram of the TV line monitor, shown in **Figure 1**, is fairly elaborate. The shaded elements are located in a programmable EPLD IC (Altera type EPM7064). A smaller version of this IC has already been used in several *Elektor Electronics* projects. Here this IC has three principal functions: it counts sync pulses, it controls the input A/D converter and RAM, and finally it looks after the conversion of the stored line back into analogue form. Let's look at the various elements of the block diagram in more detail.

CVBS LINE

The first item that the input video signal encounters is labelled 'CVBS LINE'. This is actually just a few passive components, as can be seen from the otherwise not particularly informative schematic diagram shown in **Figure 2**. The amplitude of the input signal can be adjusted with P1, while P3 adds a d.c. component to the signal. Diode D2 and capacitor C7 clamp the sync impulses to a constant d.c. level. After this processing, the signal arrives at the A/D converter IC, a National Semiconductor ADC1175CIJM. This has an input range of + 0.6 V to + 2.6 V.

VID AMP and AS

The two-stage video amplifier (T2 and T3) is also connected to the input. It raises the amplitude of the video signal by at most a factor of around 2. The input impedance has been modified to be 125 Ω instead of the usual 75 Ω, to provide satisfactory operation with input voltages lower than 1 V_{pp}. If on the other hand the input voltage is too high, overloading of the amplifier can be prevented by adjusting the level control P4. Sync pulses are clamped by D1 and C2.

The block labelled 'AS' contains a low-pass filter (R9/C8) that blocks HF components, such as the burst and chrominance signals.

T4 generates positive vertical and horizontal sync pulses with an amplitude of 5 V_{pp} (signal SHV).

Counting in windows

The SHV signal contains all the information that the programmed logic needs in order to generate the necessary clock and reset signals (XSHE, XSHV and XODD) for the three counters programmed into the EPM7064. All of these signals, by the way, pass through edge detectors which convert them into 'needle' pulses.

Pixel counter

The binary pixel counter, labelled 'ACTR', addresses the RAM for both reading and writing. It is 10 bits wide

2

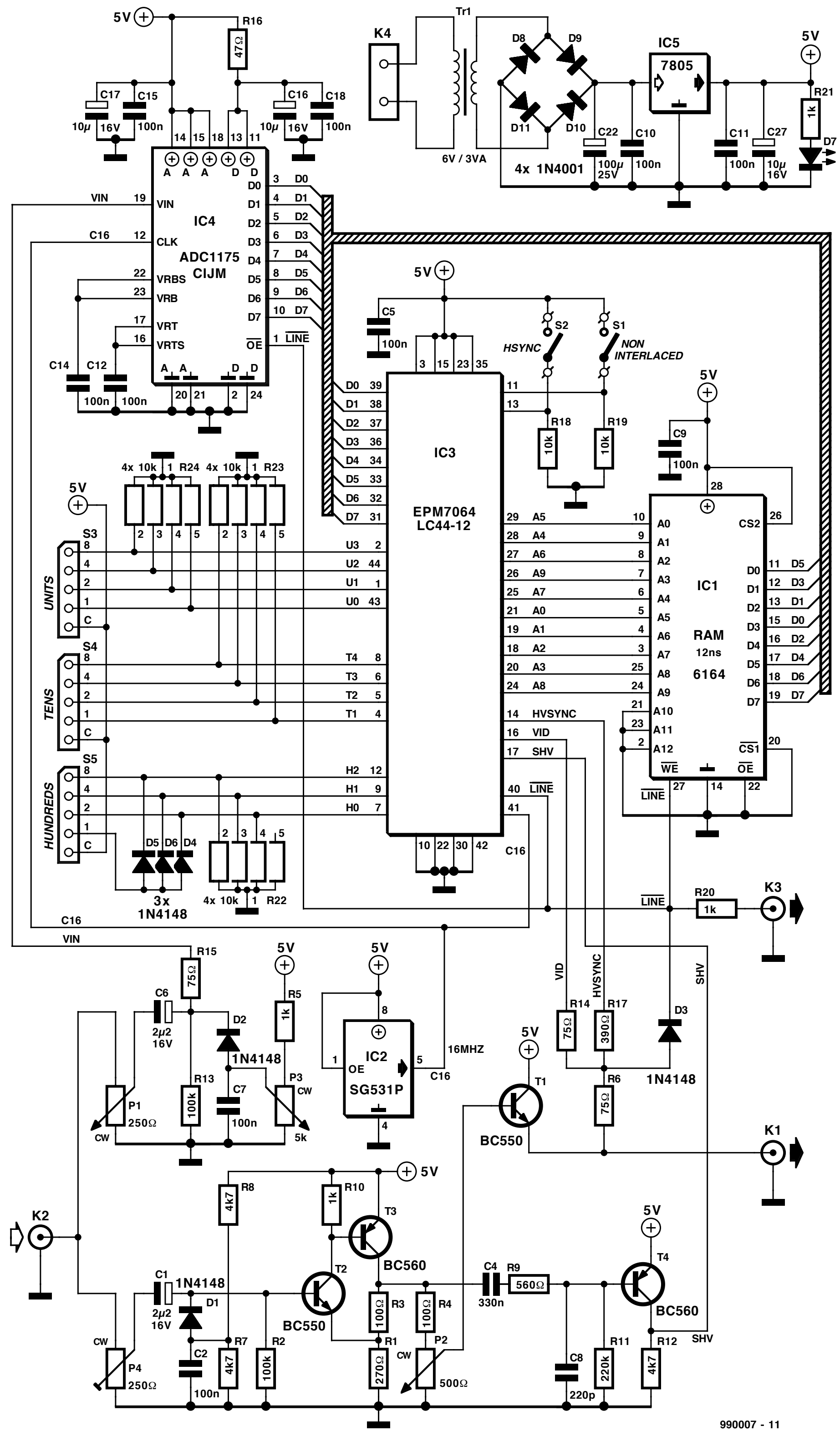


Figure 2. Schematic diagram of the enhanced TV line monitor.

and is clocked at 16 MHz. It is reset by the leading edge of the line sync pulse XSHE. For reliable synchronisation, the ACTR logic generates an 0.5-μs time window (XSHW) for the leading edge of line sync pulse. This (as well as the other time windows) acts as effective interference suppressor. Only the leading edge of the SHV signal that occurs within the XSHW window produces the XSHE signal, which resets the ACTR and also clocks the line counter.

Line counter

The line counter, labelled 'PCTR', is an 8-bit binary down counter which provides the programmed-logic D/A converter with the current line number of the input video signal. In addition, PCTR provides the two signals AVV and AVL. AVV marks the region in which the merged-in line oscillogram can be displayed (in lines 60 through 275 of the first field and 372 through 587 of the second field). AVL refers to line identification (first field = lines 24 through 311, second field = lines 336 through 623). These regions must be defined to avoid upsetting the vertical synchronisation of the television set. The PCTR counter is reset by the rising

3

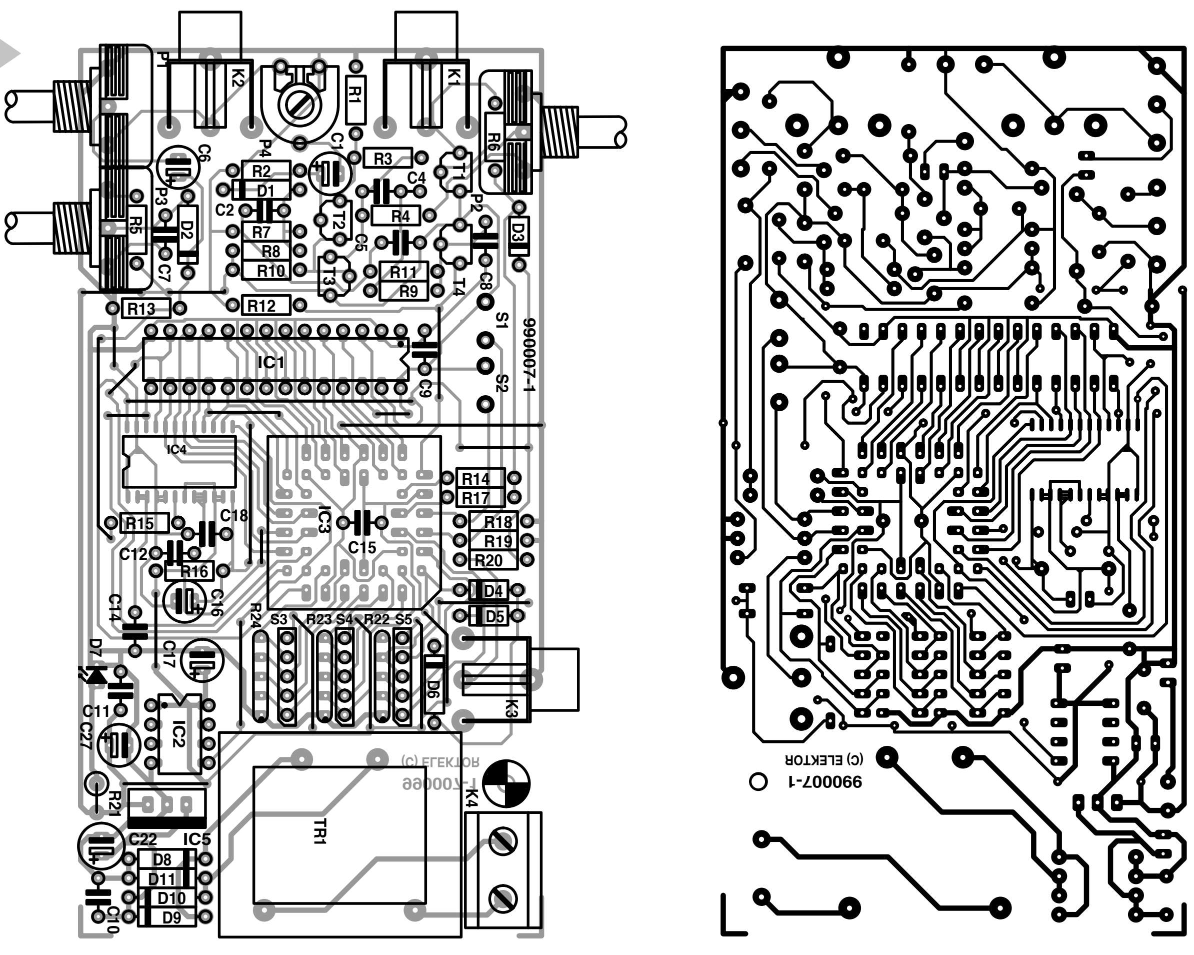


Figure 3. Everything on one board: no internal cabling is needed.

COMPONENTS LIST

Resistors:
R1 = 270Ω
R2,R13 = 100kΩ
R3,R4 = 100Ω
R5,R10,R20,R21 = 1kΩ
R6,R14,R15 = 75Ω
R7,R8,R12 = 4kΩ7
R9 = 560Ω
R11 = 220kΩ
R16 = 47Ω
R17 = 390Ω
R18,R19 = 10kΩ
R22...R24 = 4-way 10kΩ SIL array
P1 = 250Ω linear potentiometer
P2 = 500Ω linear potentiometer
P3 = 5kΩ linear potentiometer
P4 = 250Ω preset H

Capacitors:
C1,C6 = 2μF2 16V radial
C2,C5,C7,C9-C12,C14,C15,
C18 = 100nF, 5mm raster
C4 = 330nF, 5mm raster
C8 = 220pF
C16,C17,C27 = 10μF 16V radial
C22 = 100μF 25V radial

Semiconductors:
D1-D6 = 1N4148
D7 = LED, low current
D8-D11 = 1N4001
T1,T2 = BC550
T3,T4 = BC560
IC1 = 6164
IC2 = SG531P 16MHz
IC3 = EPM7064LC44-12
(order code **986523-1**)
IC4 = ADC1175-50 CIJM SMD
IC5 = 7805

Miscellaneous:
K1,K2,K3 = cinch socket, PCB mount, angled (Monacor/Monarch T-709)
S1,S2 = switch, 1 make contact
TR1 = mains transformer, 6V 3VA
S3,S4,S5 = hex thumbwheel switch (RS-components o/n 199-251, sequence C-1-2-4-8)
K4 = 2-way PCB terminal block, raster 7.5mm
Enclosure 125x70x40mm (e.g. Donau SD20)
PCB, order code **990007-1**

edge of the vertical sync pulse XSVE that falls within the timing window XSVW.

Line selector

The block labelled 'UTHCTR' is a three-digit BCD counter that is also clocked by XSHE. The ones and tens digits have four bits each, while three bits are sufficient for the hundreds digit, since the highest possible line number is only 625. This counter is reset by the leading edge of the frame sync pulse. Depending on the setting of the mode selection switch 'NIL', this is either the XSVE pulse (non-interlaced mode, lines 1 through 312) or the XODD pulse (interlaced mode, lines 1 through 625).

The UTHCTR outputs drive a comparator whose second set of inputs is connected to three thumbwheel switches. When the current line number matches the number set in the switches, the comparator generates the $\overline{\text{LINE}}$ signal. This signal passes through a flip-flop that is clocked by ACTR (and the 16-MHz clock) such that the line to be monitored is displayed starting either at the beginning of the visible horizontal line (aligned to the TV picture) or offset by a visible synchronisation pulse, depending on the position of the HSYNC switch. The signal from the flip-flop, with appropriately modified timing, is labelled '$\overline{\text{LINED}}$'. It pro-

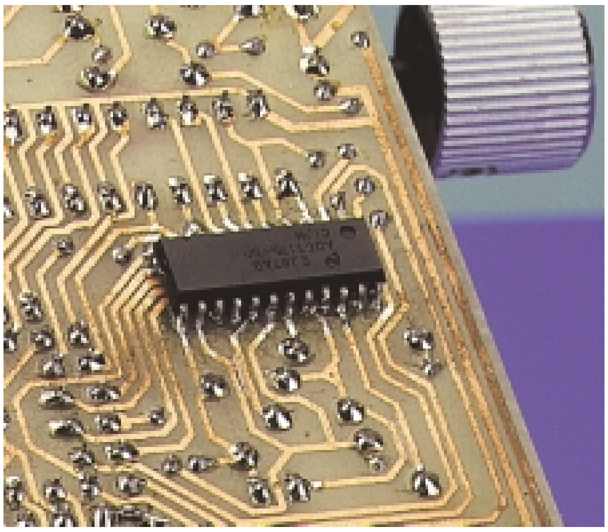


Figure 4. The A/D converter IC is mounted on the bottom (solder side) of the board.

vides the timing window for the A/D converter and video line memory.

High-speed D/A converter and video line memory

The $\overline{\text{LINED}}$ signal ensures that exactly one raster line, with a duration of 64 µs, is read, converted and stored in RAM. Since the converter is clocked at 16 MHz, 1024 samples are stored (64 µs x 16 MHz). In addition, $\overline{\text{LINED}}$ is made available at the 'LINE TRIG OUT' output socket to serve as a trigger signal for an oscilloscope.

D/A converter

The D/A converter ('DAC') is a 'home-brew' design. The standard polarity of the video signal is +CVBS, which means that the sync pulses have a low level and white video has a high level. The sync pulses are thus stored in the RAM as small binary values, while white regions are stored as large binary values. To obtain a correctly oriented display of the content of the stored line, the down counter ACTR must be used (high line numbers at the top of the screen and low line numbers at the bottom). The DAC is started in line 40 with a counter state of 255. The 8-bit comparator (DAC) compares the counter state for each raster line with the values provided by the line memory (MEM), which in turn is driven by ACTR in a periodic manner. It assigns the state of the PCTR counter to each digitized voltage value. For example, when the counter state is 155 (corresponding to raster line 140), every stored sample with a value of 155 generates a pixel in the VID signal. If AVV and AVL are both active, this forces the video signal to the white level. The line content diagram thus becomes part of the output signal.

Output signals

The output signal CVBS + VID OUT consists of the combination of the reproduced line content diagram, the CVBS signal (amplified and buffered, with adjustable amplitude) and the combined sync signal SHV. The blend of the CVBS and line content signals can be continuously adjusted with P3, starting with a colour CVBS picture with a faint oscillogram, passing through monochrome CVBS and ending with a display of the oscillogram alone without the CVBS image. The selected line is shown as a white line within the AVL region for identification.

All on one board

After this somewhat roundabout journey through the logic of the TV line monitor, we can turn our attention to its construction. As can readily be seen from **Figure 3**, all components are mounted on one single-sided printed circuit board, including the power transformer, jacks and presets. Note however that if you want to use BNC sockets instead of the specified Cinch sockets, you will have to make a few additional short cables.

Installing the components on the board is somewhat tricky, due in part to the 22 wire jumpers. Since some of these run very close to other components, you should use fine insulated wire for all of them. Once all the jumpers have been soldered in place, mount the SMD IC before any of the other components. After this, continue in the usual manner with the low-profile components such as resistors, diodes and capacitors, and finish off with the transformer, switches S1 and S2, the jacks, presets and so on. Mount the switches elevated above the board, so that their levers will later protrude conveniently from the enclosure. By the way, C15 should be mounted either flat inside the socket or on the back (solder side) of the board. The ICs as well as the thumbwheel switches may be mounted in sockets. When soldering, take care to avoid making any solder bridges between the tracks or solder pads, which are very close together is some places.

Before putting the TV line monitor into use, you should carefully inspect the finished circuit board. Once you have verified that all polarized components (including the thumbwheel switches!) have been correctly installed, attach the mains cable to K4 and secure it with a suitable strain relief. Then attach the circuit board to the lower part of the enclosure (one screw only) and press the upper part of the enclosure into place (after first making the necessary holes and cut-aways!). The two switches S1 and S2 can be left free or attached only loosely.

Practical aspects

Let's review the inputs, outputs and controls of the TV line monitor. It has a video signal input (K2, CVBS IN), which can accept a signal from a video recorder, PC video card, satellite receiver or other video source. Adjust P4 so that the strongest available video signal does not overdrive the first video amplifier stage (T2). P1 controls the amplitude of the line content oscillogram, while P2 controls its location on the screen.

The output signal leaves the TV line monitor via K2. On the connected monitor (TV), you will see the input signal (picture), a white line (the selected line) and an oscillogram of the selected line. The relative strengths of these signal can be adjusted with P3.

If you want to observe the selected line on an oscilloscope, connect the CVBS signal to the Y input and the line monitor K3 output to the external trigger input of the oscilloscope.

Since the picture contents of adjacent lines in the first and second fields are often nearly identical, the number of sampled pixels can be doubled by displaying both lines at once. To do so, set switch S1 to non-interlaced (2 x 312 lines). This operating mode also allows non-interlaced television signals to be processed, such as that produced by the Elektor Video Test Chart Generator (September and October 1996 issues). If you wish to observe only one line per frame (1 x 625 lines), S1 must be set to interlaced mode.

The function of switch S2 can best be demonstrated by trying it out with the TV line monitor in operation: the oscillogram in the CVBS output signal either includes a sync pulse, or it doesn't.

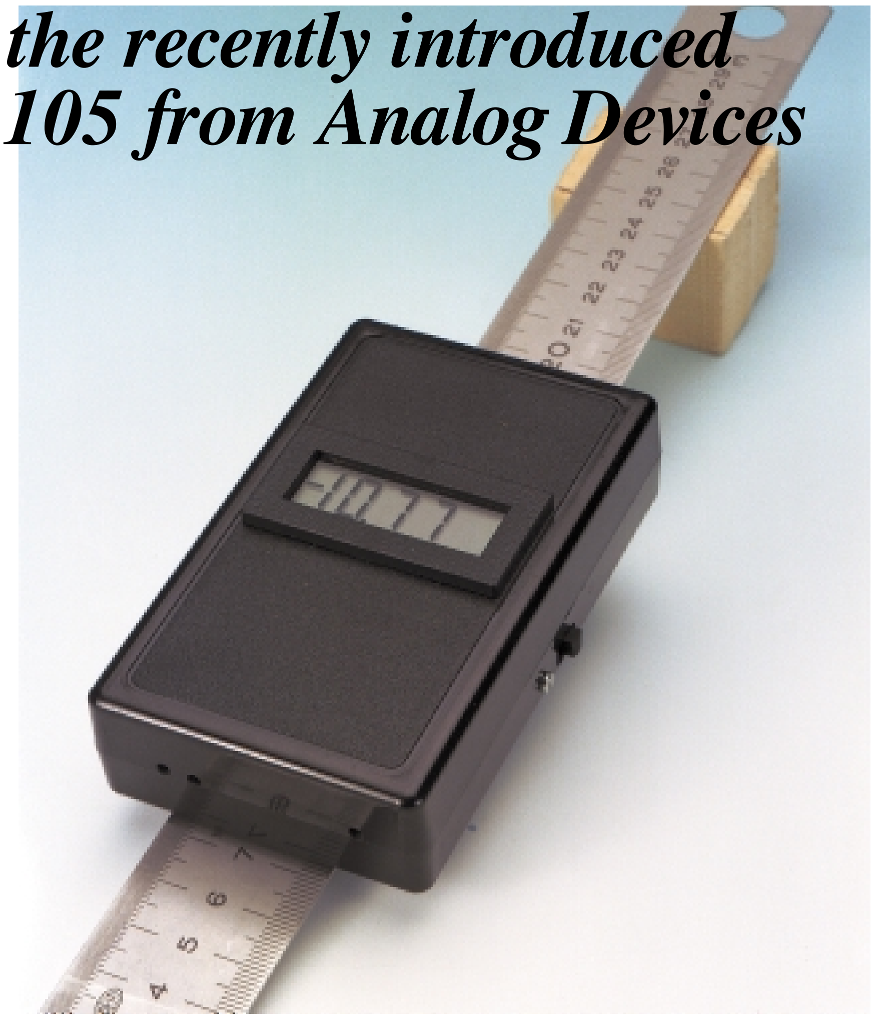
(99007-1)

gradient* meter

based on the recently introduced Type ADXL105 from Analog Devices

An integrated accelerometer circuit such as the ADXL105 from Analog Devices, although intended primarily for measuring acceleration and deceleration, may readily be used for gauging gradients*. If the circuit is to be used out of doors, its temperature compensation must be adapted accordingly.

Design by P. Porcelijn



INTRODUCTION

The Type ADXL05 accelerometer IC from Analog Devices was introduced in a relevant Application Note in the April 1997 issue of *Elektor Electronics*. Just over a year later, the device was used as the basis of an 'electronic accelerometer' (June 1998 issue of *EE*).

From the properties of the circuit it is clear that the device may readily be used for measuring gradients. When this is to be done with some accuracy in a motor vehicle or on a bicycle, it is, of course, necessary to maintain a constant speed.

There are two further matters to be resolved before gradients can be gauged with an accelerometer: the choice of a suitable display, and adding suitable temperature compensation.

The question of a display is relatively easily satisfied. This is achieved by obtaining a direct indication of the percentage gradient from the relationship between the acceleration of free fall, g, the gradient, p, in per cent, and adjusting the amplification of the circuit accordingly. From the discussion in the box 'Principle of measurement' it will be seen that, within narrow tolerances, the output voltage of the sensor is virtually directly proportional to the gradient percentage.

Solving the matter of temperature compensation is rather more tricky. Compensation is necessary because when the device was tested for use as a gradient gauge on a bicycle it was found that temperature variations caused such inaccurate measurements as to render them unusable.

Fortunately, Analog Devices recently upgraded the ADXL05 to the Type ADXL105, which is provided with an internal temperature sensor. The addition of a few simple electronic circuits to the sensor gives the upgraded device very efficient temperature compensation.

CIRCUIT DESCRIPTION

The circuit diagram is shown in **Figure 1**. It will be seen at a glance that, like the earlier accelerometer, the gradient meter is battery operated. Also, the display is again formed by a Type DPM951 digital voltmeter (DVM)

* *Also, especially in USA and Canada, grade: a part of a road or railway that slopes upwards or downwards, an incline. It also indicates the measure of such an incline (normally as a percentage).*

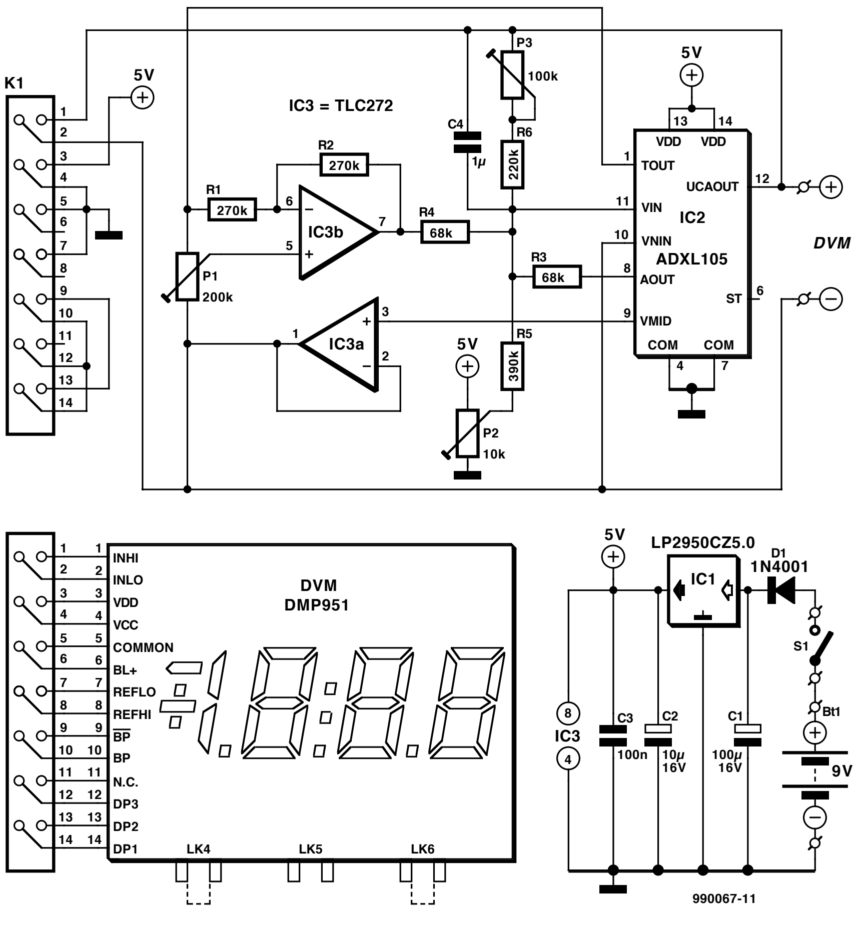


Figure 1. Other than the accelerometer IC and the digital voltmeter (DVM) module, the electronics is limited to two operational amplifiers and some passive components.

linked to the circuit via box header K_1.

It will also be seen that, whereas the accelerometer comprises just the ADXL05 chip and the DVM module, the gradient meter has two additional operational amplifiers (op amps), one of which is needed for the temperature compensation circuit.

According to the manufacturer's specification, the temperature drift of the ADXL105 is about 1 mV °C^{-1}, that is, roughly 60 mV over the entire temperature range. Since the error in the gradient meter is amplified ×4, the overall drift works out at 4 mV °C^{-1}, which, as the polarity of the temperature coefficient is not known, will lead to unusable results.

The integral temperature sensor of the ADXL105 provides a temperature-dependent voltage, U_t, at output T_{out} (pin 1) of

$$U_t = 2.5 + 8\times10^{-3}\times(t-25) \quad [\mathrm{V}]$$

with a tolerance of ± 0.1 V (t is the ambient temperature).

The output of the sensor is modified to a usable value with the aid of a compensating circuit based on op amp IC_{3b}. This circuit enables the temperature coefficient to be set between –8 mV °C^{-1} and + 8 mV °C^{-1} with P_1. In principle, a passive circuit could have been used, but that would have made the calibration fairly tedious.

Unfortunately, the integral potential divider in the ADXL105 has an output impedance of 10 kΩ. This is rather high for the present application, particularly since the output voltage is also used for the DVM module. This small problem is remedied by a buffer, op amp IC_{3a}.

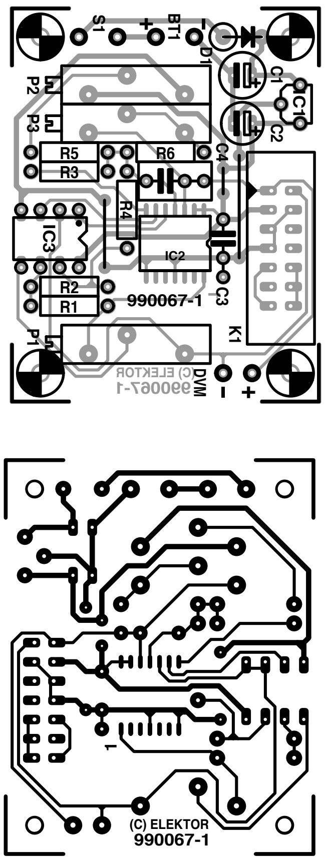
OFFSET AND GAIN

The passive components in the circuit diagram not yet dealt with form part of the offset compensation and requisite amplification circuits.

The offset error of the ADXL is not insignificant: it may be as high as ± 625 mV. Fortunately, compensating for this is easily achieved by injecting a variable direct current into V_{IN} (pin 11) via resistor R_5 and preset P_2. At unity gain, the range of this preset is ± 870 mV.

The input range of the specified DVM module is 200 mV. Assuming that the maximum gradient (grade) to be

Figure 2. Printed-circuit board for the gradient meter which is, however, not available ready made.



Parts list

Resistors:
R_1, R_2 = 270 kΩ
R_3, R_4 = 68 kΩ
R_5 = 390 kΩ
R_6 = 220 kΩ
P_1 = 200 kΩ, 10-turn preset
P_2 = 10 kΩ, 10-turn preset
P_3 = 100 kΩ, 10-turn preset

Capacitors:
C_1 = 100 μF, 16 V, radial
C_2 = 10 μF, 16 V, radial
C_3 = 0.1 μF, ceramic
C_4 = 1 μF, metallized polyester

Semiconductors:
D_1 = 1N4001

Integrated circuits:
IC_1 = LP2950CZ5.0
IC_2 = ADXL105JQC
IC_3 = TLC272CP

Miscellaneous:
BT_1 = 9 V battery with connecting clips
S_1 = slide switch, 1 make contact
K_1 = 14-way box header
DVM module as appropriate
Enclosure as appropriate

principle of measurement

Owing to the earth's gravity, any freely suspended object is subject to an acceleration of free fall, g, which, near the earth's surface, is equivalent to 9.80665 m s^{-2}. If the object is on a gradient of p%, its acceleration, g_s is

$$g_s = g \sin\alpha, \qquad [1]$$

where α is the gradient, and

$$\sin\alpha = p/100/\sqrt{\{1 + (p/100)^2\}} \qquad [2]$$

If p/100<< 1, this equation simplifies to

$$\sin\alpha \approx p/100. \qquad [3]$$

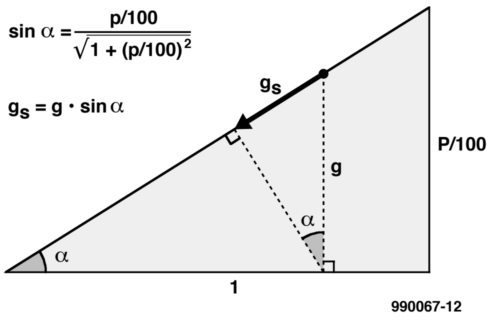
When the latter equation is used, a small error occurs :

0.5% when p= 10%
2% when p= 20%

Since the error remains within acceptable limits, equation [3] is perfectly all right for practical use.
Combining equations [1] and [3] gives the following relationship between acceleration and gradient:

$$p = (g_s/g) \times 100\%$$

This equation shows that if a small error is acceptable, the output voltage of the accelerometer IC is directly proportional to the gradient.



$\sin\alpha = \frac{p/100}{\sqrt{1+(p/100)^2}}$
$g_s = g \cdot \sin\alpha$
g_s
g
P/100
α
α
1
990067-12

traversed is 20% (a steeper incline is seldom encountered on normal roads), these two quantities can be equated very satisfactorily.

Since the sensitivity of the accelerometer IC is 0.25 V g^{-1}, its output voltage in volts is a quarter of the maximum gradient in per cent. This means that all that needs to be done is to make the output voltage of IC_2 in volts equal to the gradient in per cent. A simple calculation shows that this requires an amplification of ×4, which is set with resistors R_3 and R_6. Any tolerances are compensated with preset P_3 in series with resistor R_6. Any interference is minimized by capacitor C_4, whose value is chosen such that the reaction of the circuit to varying circumstances is not unduly delayed.

CONSTRUCTION

Building the gradient meter on the printed-circuit board in **Figure 2** is straightforward. Note, however, that this board is not available ready-made. Care should be taken to ensure that the sensor is soldered level on the board, since for correct operation it must be parallel with the board.

The voltmeter module is linked to the circuit via box header K_1. If a module other than specified is used, it may be possible to omit K_1 and solder the module terminals directly to 'DVM' on the board.

The completed board fits nicely in the specified enclosure. Note that the fixing bosses in the enclosure are not used: the board is simply screwed directly to the bottom. Good care should be taken because it is a tight fitting.

Before the board is fitted in the case, drill three holes in the top lid so that presets P_1–P_3 are accessible from the outside.

POWER SUPPLY

The DVM module is powered by a 9 V dry or rechargeable battery. The battery voltage is reduced to 5 V by voltage regulator IC_1. Since the accuracy of the voltmeter is wholly dependent on the supply voltage, the regulator should be a type with very low temperature coefficient: better than 150 ppm °C^{-1}. The type specified is also a low-drop model which makes it ideally suited for use in battery operated equipment.

The gradient meter draws a current of about 6 mA. Diode D_1 guards against incorrect polarity of the supply lines.

CALIBRATION

Offset & temperature compensation

1. Place the gradient meter in a level position (check with a spirit level) in a constant ambient temperature of 20 °C and leave it to settle down for about half an hour.
2. Set P_3 to the centre of its travel. Connect a digital voltmeter (direct millivoltage range) between '–DVM' and pin 7 of IC_3. Adjust P_1 until the meter reading is exactly 0.
3. Turn P_2 until the DVM module also reads exactly 0.
4. Heat the gradient meter with, say, a hair dryer, whereupon the reading on the DVM will probably change. Readjust P_1 until the module again reads exactly 0.
5. Let the gradient meter cool off for about half an hour and, if necessary, readjust P_2 until the DVM module reads exactly 0.
6. Repeat steps 4 and 5 until a satisfactory temperature compensation is obtained.

Amplification

The amplification should be set with the gradient meter on a gradient (grade). The discussion in the box shows that the approximation $\sin\alpha \approx p/100$ has an error that increases in direct proportion with α. For instance,

at $p \approx 0\%$, the error is 0%,
at $p \approx 10\%$, the error is 1%,
and at $p \approx 20\%$, the error is 2%.

The error may be minimized by calibrating at about the centre of the measuring range, rather than at the extremes. A simple calculation shows that the maximum error can be brought down to 1% if calibration takes place at a gradient of about ×1/1.4 (0.712) the maximum. So, if the maximum is taken as 20%, calibration should take place at a gradient of 14%.

The gradient is readily obtained by fixing (adhesive tape) the gradient meter at the centre of a 1 metre (39.3 in) long strip of straight wood. A gradient of 14% is obtained by placing one end of the strip of wood 14 cm higher than the other end. When this is done, adjust P_3 until the DVM module reads the percentage gradient of 14%.

[990067]

BASIC Stamp programming course (3)

part 3: BASIC programming

In this instalment we'll introduce BASIC Stamp branching and EEPROM access commands used to make your BOE-Bot follow a pre-determined path. A piezo-speaker will provide a feedback mechanism to identify the location within your program. We'll also introduce a mobile temperature logger as an optional project to help you learn synchronous serial communication.

Attention. An important Correction regarding the Elektor Electronics BOE-Bot development board appears elsewhere in this issue.

By Chuck Schoeffler, Ph. D., Ken Gracey and Russ Miller

13

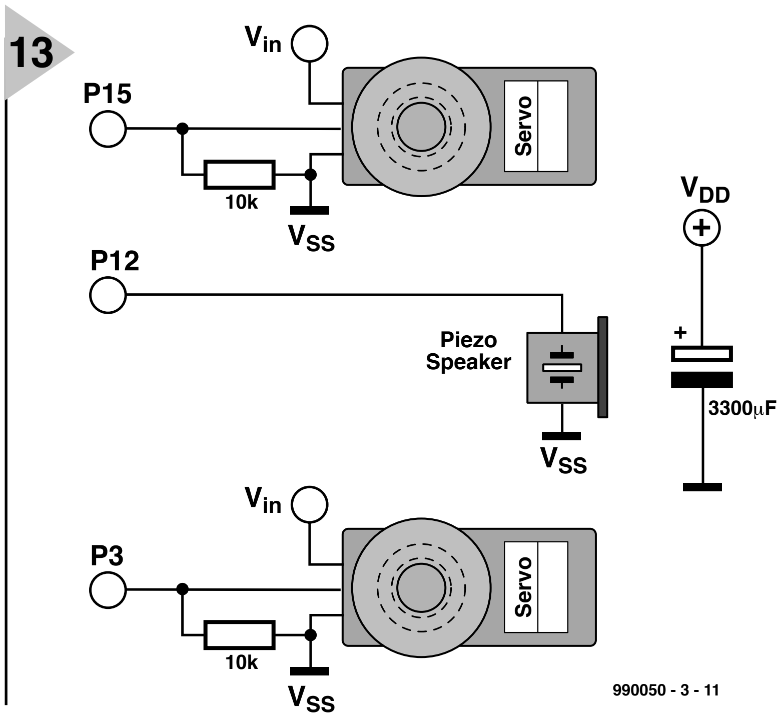


Figure 13. Basic control and feedback schematic.

Parts List.

1	Complete BOE-Bot (robot built on Elektor Electronics Board of Education)
1	Piezospeaker
1	3300μF capacitor
2	10kΩ resistors (optional in circuit)

MOVEMENT USING SUBROUTINES AND MEMORY

Movement is one of the most distinctive features of robots, and it is also an ideal way to learn how to structure and write a simple PBASIC program. This experiment is all about BASIC programming as it pertains to BOE-Bot movement without sensor input. Structuring your program so the BOE-Bot moves as you intend requires an understanding of how to call subroutines, read movement patterns from an EEPROM, know how far to travel using a for-next loop, and how to get back to your starting point (physically, and within your source code). The parts we will be using are listed in **the Parts List**. The complete schematic for programs used in this column is shown in **Figure 13**.

Review of servo control

Servos are closed loop devices, and they are constantly comparing their commanded position (from the BASIC Stamp's pulsout command) to their actual position (proportional to the resistance of a potentiometer mechanically linked to the shaft). If there is more than a small difference between the two, the servo's electronics will turn the motor to eliminate the error.

We modified the servo's potentiometer shaft until the gears stopped moving when the BASIC Stamp sent a 1500 μs pulse. A pulsout value of 750 is equal to 1500 μs (the command operates in units of two microseconds). A value larger than 750 will turn the servo clock-wise, and a value less than 750 will turn it counter-clockwise. A value very close to 750, like 760, will cause the servo to turn very slowly. **Figure 14** is a timing diagram of the pulsewidth modulation.

A for-next loop can be used to see how different pulsewidths affect the servo's speed. Stand your BOE-Bot on its front or place an object underneath it to keep it from rolling away. Download **Program Listing 1** to your BASIC Stamp. **Figure 15** is a graph of pulsewidth compared to revolutions per minute using the Futaba S-148 servo.

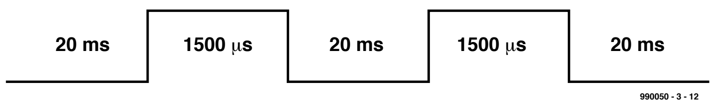
14


Figure 14. Servo control using pulsewidth modulation.

```
'Program Listing 1
left_servo     con     15
right_servo    con     3
x var word
pause 2000
start:
for x = 650 to 850                          'begin of routine
     pulsout left_servo,x                   'pulse width of 1500 us
     pulsout right_servo,1500-x             'pulse width of 1500 us
     pause 20                               'pause for 20 ms
next
```

```
'Program Listing 2
Hz var word
for Hz = 1 to 4000 step 1000
  freqout 12,70,Hz,4000-Hz
'generate two 70 ms tones on P12
next
```

Sound feedback

The BASIC Stamp's freqout command can be used to add sound feedback to your BOE-Bot. Like all PBASIC commands, it has a particular syntax that must be followed to make it work. To hear the speaker, download the following code to your BASIC Stamp:

```
freqout 12,750,2000
'750 ms 2000 Hz tone on P12
```

For a more "robotic" sound try the example in **Program Listing 2**.

This routine begins by declaring Hz as a word variable, a number between 0 and 65,536. The loop executes a total of four times ((4000-1)/1000), generating two frequencies at once on P12. The first frequency is increasing from 1 to 4000 Hz while the second frequency is decreasing from 4000 to 1 Hz. Sounds like this could be added throughout your program.

Goto statement

Normally, PBASIC programs execute instructions line by line. The goto command causes the BASIC Stamp to jump to a named place somewhere else in the program. It can be either forward or backward in the program. The syntax is quite simple.

```
goto  forward'jump to the
forward routine
```

Gosub is a close relative of Goto

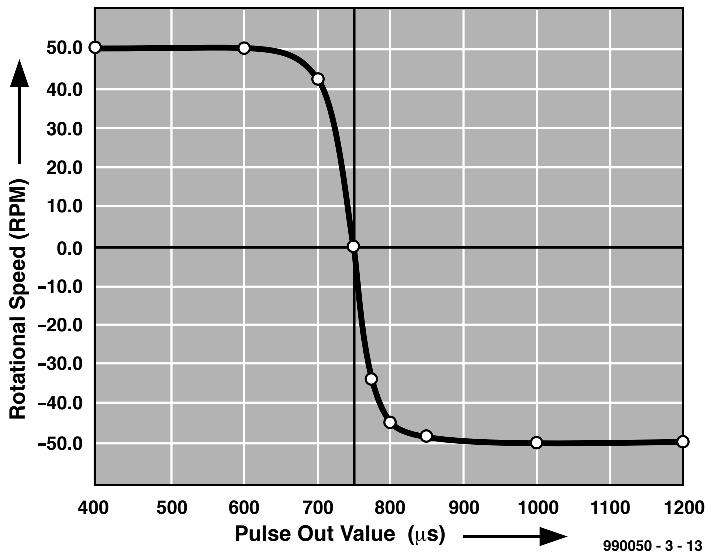
The gosub (Goto Subroutine) statement also causes the program execution to jump somewhere else, but the line after the gosub is remembered so that the program can automatically go back and continue where it left off. This lets us easily reuse sections of the program. The following example illustrates the gosub command.

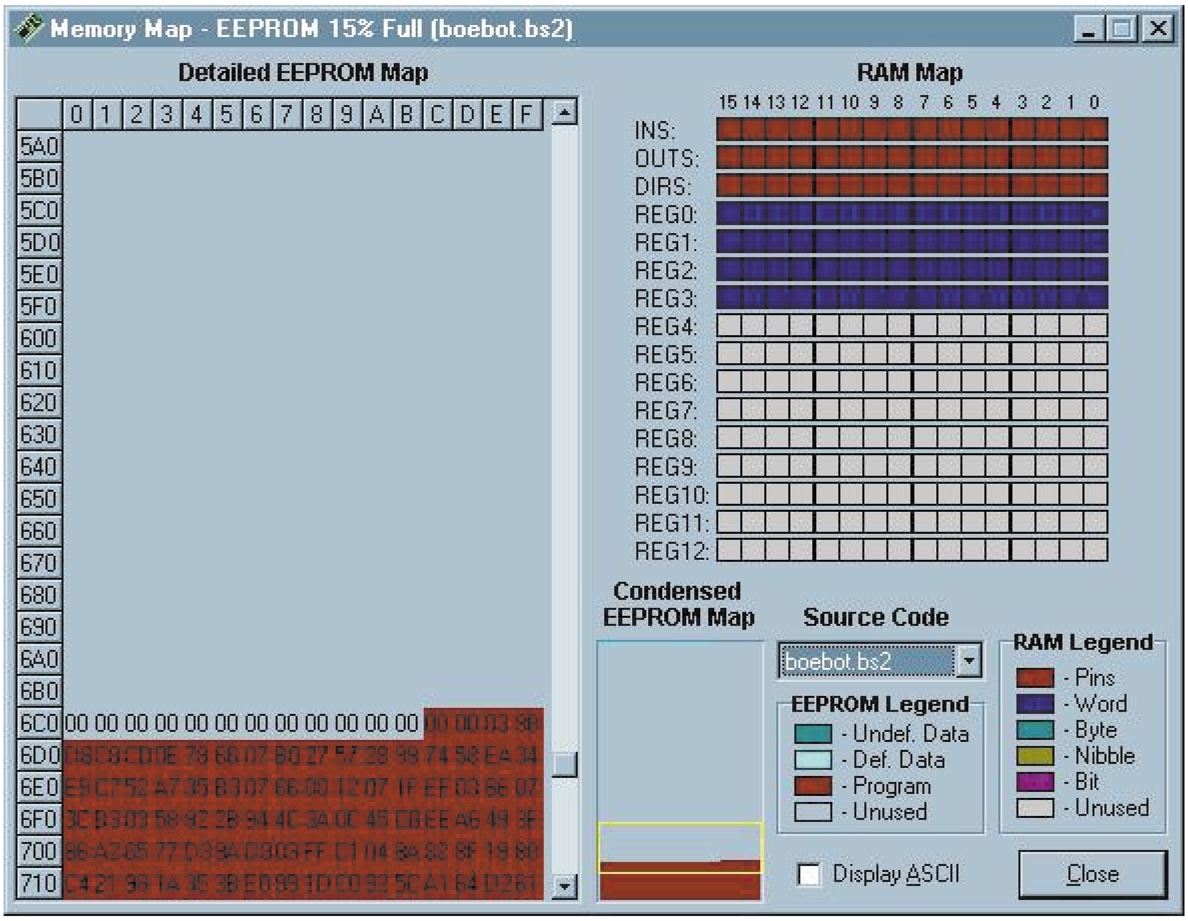
```
gosub right
pause 1000
gosub right
return

right:
for x=1 to 18
   pulsout left_servo,650
   pulsout right_servo,650
pause 20
```

This example shows the right routine being executed twice with a one-second pause. The gosub commands may also be nested up to four deep, so that

Figure 15. Pulsewidth versus RPM using the Futaba S-148.

15




Memory Map - EEPROM 15% Full (boebot.bs2)

Detailed EEPROM Map

```
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
5A0
5B0
5C0
5D0
5E0
5F0
600
610
620
630
640
650
660
670
680
690
6A0
6B0
6C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 98
6D0 D8 C8 CD 0E 78 66 07 B0 27 57 28 99 74 58 EA 34
6E0 E9 C7 52 A7 35 B3 07 66 00 12 07 1F EF 03 66 07
6F0 3C B3 03 58 92 2B 94 4C 3A 0C 45 CB EE A6 49 3F
700 86 A2 65 77 D3 8A D3 03 FF C1 04 8A 82 8F 19 80
710 C4 21 93 1A 35 3B E0 99 1D C0 92 5C A1 64 D2 61
```

RAM Map

```
        15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
INS:
OUTS:
DIRS:
REG0:
REG1:
REG2:
REG3:
REG4:
REG5:
REG6:
REG7:
REG8:
REG9:
REG10:
REG11:
REG12:
```

Condensed EEPROM Map

Source Code: boebot.bs2

EEPROM Legend
- 🟦 - Undef. Data
- ⬜ - Def. Data
- 🟥 - Program
- ⬜ - Unused

RAM Legend
- 🟥 - Pins
- 🟪 - Word
- 🟦 - Byte
- 🟨 - Nibble
- 🟪 - Bit
- ⬜ - Unused

☐ Display ASCII

Close

Figure 16. EEPROM memory map.

each return takes the program back to the instruction after the most recent gosub.

Using the Data statement and EEPROM to store movements

The BASIC Stamp has a 2K EEPROM that is used for program storage (which builds downward from address 2047) and data storage (stores in the opposite direction — from address 0 toward 2047). If the data collides with your program the source code will not execute properly. Each location is a byte. This isn't enough memory to build a complex environmental data logger, but it's certainly enough space to store bytes of information you'd like to use in a program.

The BASIC Stamp's EEPROM is different from RAM variable storage in several aspects:

- EEPROM takes more time to store a value, sometimes up to several milliseconds.
- EEPROM can accept a finite number of write cycles, around 10 million writes to be exact (RAM has unlimited read/write capabilities).
- Primary function of the EEPROM is

```
'Program Listing 3
'Boe-Bot Program for Roaming, Light, and Sound
'Define Variables and Constants
'————————————————————————————
x          var     word              'loop counter for pulsout
position           var     word      'EEPROM address counter
direction          var     word      'value stored in EEPROM
Hz         var     word              'frequency variable
right_servo        con     3         'right servo on P3
left_servo         con     15        'left servo on P15
speed              con     40        'added or subtracted value

'————————————————————————————
'Programmed Movement Patterns
'————————————————————————————
data "FRFRFRBBTFE"                   'store movements

'————————————————————————————
'Main Program
'————————————————————————————
position=0                           'start at EEPROM cell 0
move:                                'main loop
read position,direction              'read direction command
position=position+1                  'increment to next cell
  if direction="E" then quit         'Decide which action to take
  if direction="F" then forward'by matching command letter
  if direction="R" then right
  if direction="L" then left
  if direction="B" then backward
  if direction="T" then turn_around
goto move                            'repeat until E is seen

'————————————————————————————
'Sound Routines
'————————————————————————————
forward_sound:
for Hz = 1 to 4000 step 1000
  freqout 12,70,Hz,4000-Hz
next
```

Approximating distance of travel

It's easy to approximate distance travel by calculating the circumference of a wheel and estimating the pulsewidth timing loops executed by your PBASIC code. First determine wheel circumference.

circumference = pi multiplied by wheel diameter
circumference = 3.14159 x 6.67 cm = 21 cm

Knowing the rotational speed of different pulse widths will allow you to determine a specific travel distance. For example, a `pulsout` *command of 850 causes the servo to turn at about 50 revolutions per minute (RPM) or 0.83 revolutions/sec. Therefore the speed of the robot will be about:*

21 cm/revolution x 0.83 revolutions/sec = 17.5 cm/s

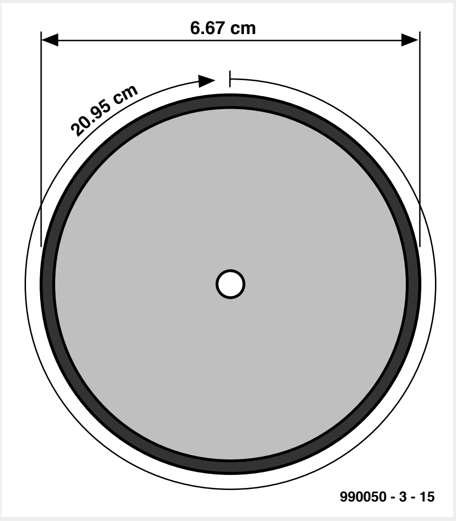
To travel 100 cm, the BOE-Bot would have to travel for:

100 cm / 17.5 cm/s = about 5.7 seconds

Since each servo pulse takes about 1.5 ms and there is a 20 ms pause in the loop, each loop will take about 23 ms (1.5+ 1.5+ 20), or 0.023 seconds to execute. A total of 247 loops is required to travel 100 cm. You may have to adjust the for...next loop for your particular servo.

5.7 sec / 0.023 sec/loop = 247 loops

```
.
.
forward:
for x=1 to 247
  pulsout left_servo, 650
  pulsout right_servo, 850
  pause 20
next
```



```
return

back_sound:
for Hz = 4000 to 6000 step 1000
  freqout 12,70,Hz,Hz-400
next
return

right_sound:
  freqout 8,800,2500
return

left_sound:
  freqout 8,800,4500
return

' ─────────────────────────────
' Movements
' ─────────────────────────────
forward:
gosub forward_sound
for x=1 to 60
  pulsout left_servo,750-speed
  pulsout right_servo,750+speed
pause 20
next
goto move

backward:
gosub back_sound
for x=1 to 60
  pulsout left_servo,750+speed
  pulsout right_servo,750-speed
pause 20
next
goto move

right:
high 0
gosub right_sound

for x=1 to 18
  pulsout left_servo,750-speed
  pulsout right_servo,750-speed
pause 20
next
low 0
goto move

left:
high 14
gosub left_sound

for x=1 to 18
  pulsout left_servo,750+speed
  pulsout right_servo,750+speed
pause 20
next
low 14
goto move

turn_around:
for x=1 to 30
  pulsout left_servo,850
  pulsout right_servo,850
pause 20
next
goto move

quit:
end
' ─────────────────────────────
```

to store programs; data is stored in leftover space.

Three commands are used to access the EEPROM: `data`, `read`, and `write`. The data stored in EEPROM builds from the upper left-hand corner (position 0,0) and fills downward in a left to right fashion, by row. The source code builds from lower right-hand corner (position 16,128) and builds upward by row, right to left. If the two collide, the BASIC Stamp will not execute the program properly. This is shown in **Figure 16**.

The EEPROM memory map is accessed within the BASIC Stamp Windows editor under Run/Memory Map.

The syntax for the `read` command is shown below. The `write` command uses the same syntax.

```
write 0,100
  'write 100 into eeprom byte 0
read 0,x
  'read eeprom byte 0 and store
                     value in x
debug dec ? x
    'display value on PC screen
```

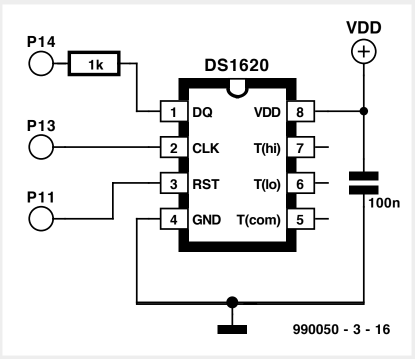
COMBINE THE CONCEPTS

Program Listing 3 brings all of these concepts together: sound, movement, and speed. In order to make effective use of the speed, you will need to iden-

Extra Topic: Synchronous serial communication with the DS1620 Digital Thermometer

Sending bytes of data with a BASIC Stamp is handled by the **shiftin** *and* **shiftout** *commands. If you are ready to learn about serial communication, try this project on your Board of Education.*
The following additional parts are required:

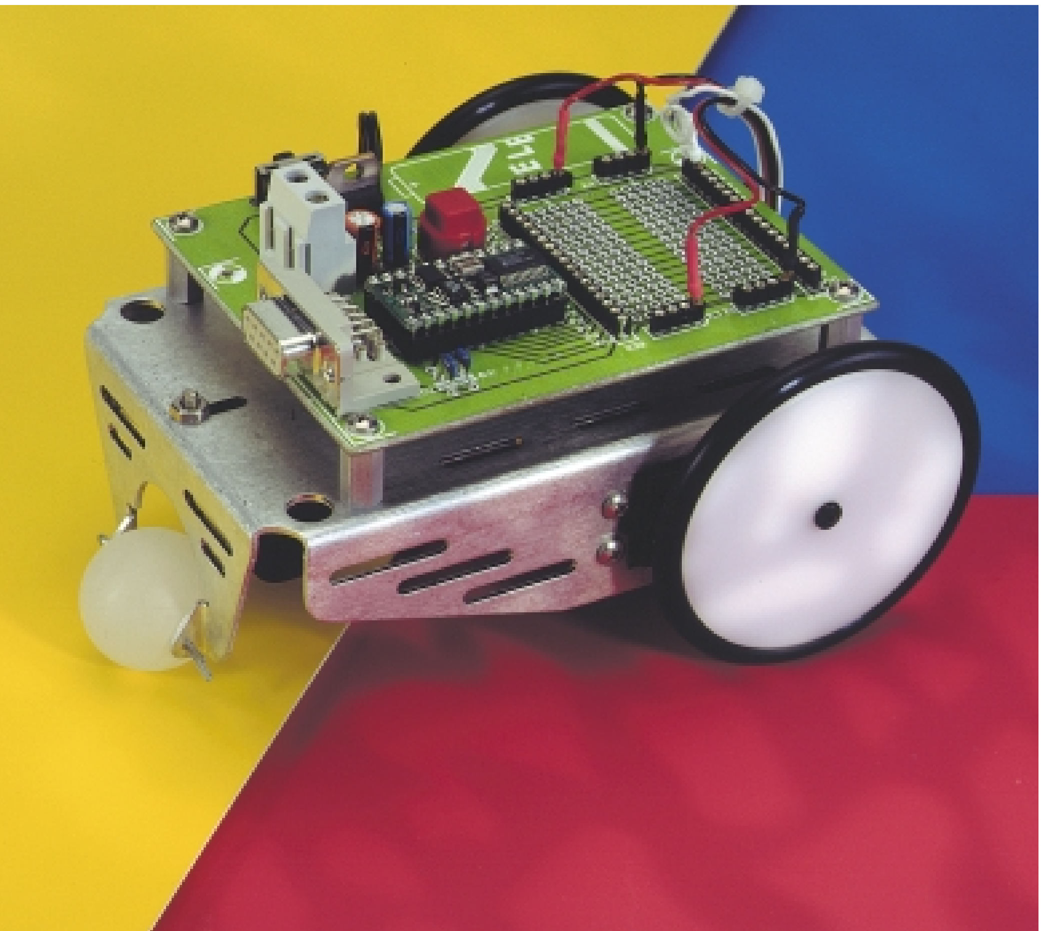
One DS1620 Temperature Sensor
One 1 kΩ resistor
One 0.1 μF capacitor



```
x       var byte                          ‘ define a general
purpose variable, byte
degC  var byte                            ‘ define a variable to hold degrees Celsius
                                 ‘ note: DS1620 has been preprogrammed for mode 2.
outs=%0000000000000000                    ‘ define the initial state of all pins
     ‘fedcba9876543210
dirs=%1111111111111111                    ‘ as low outputs

freqout 0,20,3800                         ‘ beep to signal that it is running
high 11                                   ‘ select the DS1620
  shiftout 14,11,lsbfirst,[238]           ‘ send the ”start convertions” command
low 11                                    ‘ do the command
  loop:                                   ‘ going to display once per second
  high 11                                 ‘ select the DS1620
  shiftout 14,13,lsbfirst,[170]           ‘ send the ”get data” command
  shiftin 14,13,lsbpre,[x]                ‘ get the data
  low 11                                  ‘ end the command
  degC=x/2                                ‘ convert the data to degrees C
  debug ? degC                            ‘ show the result on the PC screen
  pause 1000                              ‘ 1 second pause
goto loop                                 ‘ read & display temperature
```

Once the program is working try to use the write command to save temperature readings to the EEPROM, and read to receive them back and display on a debug terminal.



tify the exact centre position of your servo. Chances are although we started with 750 (1500 μs) the servo may have wandered to a slightly different value. Program Listing 1 may be run to identify these values. At higher speeds (speed constant around 40) this is not as visible, but at slower speeds the BOE-Bot will slowly move sideways if the center position is not exactly 750. Change values, movement patterns, and place the sound routines in different sections of the program. This program is also available from the downloads section in
http://www.stampsinclass.com.

(990050-3)

battery charge/refresh station (2)

construction, adjustment and practical use

Last month's article, which describes the features and circuit diagram of the battery charge/refresh station, clearly suggests that completing such a project, with peak charging currents of up to 8 A, is a bit more demanding than for a device with fewer automatic features and relatively lower performance. The construction information given here is thus especially important if you wish to obtain a good result. In addition to descriptions of the construction and the microprocessor-controlled, menu-driven alignment procedures, you will find many helpful, practical tips regarding battery charging and maintenance.

Design by N. Bechtloff & G. Brenner
(Conrad Technology Center, CTC)

Before starting, you should understand that the battery charge/refresh station is not a suitable project for beginners. In addition to the relatively complex electronics and several alignment procedures, the relatively high output current (8 A peak maximum!) of this high-performance charger can result in significant damage in case of a short circuit. However, if you have already successfully built a high-power amplifier or a laboratory power supply, you should not have any difficulty with this power project, as long as you work carefully.

CONSTRUCTION HINTS

The charge/refresh station is built using two single-sided printed circuit boards, as shown in **Figures 4 and 5**. These are supplied as a set and thus have a single order number in the components list. One disadvantage of using single-sided boards is a relatively large number of wire jumpers. It is best to begin by inserting these jumpers, and to carefully check your results against the component layout diagram to be sure that there are in fact seven on the smaller board and eight on the larger one. The pins for jumper JP1 can best be placed on the solder side of the front panel board, so that the jumper is accessible after the LCD module has been mounted on the component side. IC4 must be mounted flat on the larger board, so that its case does not push against the front panel. Pushbutton S3, switch S1 and LED D6 should be provisionally connected to the circuit board to start with. After the unit has been tested and aligned, they can be mounted in their final positions on the front panel and connected to the circuit board using light-duty flexible wire. The socket for the microcontroller can initially be left empty, and the LCD module need not be plugged into the socket strips right away, since neither of these components is necessary for the first two alignment procedures (fan voltage and temperature monitoring).

The temperature monitoring circuitry is optional and need not necessarily be installed. If you do not wish to monitor the temperature of the heatsink or the battery, you can eliminate not only the NTC sensor but also resistors R54, R56, R80 and R82 (R81 must always be used). The NTC sensor should have a cold resistance between 500 and 1000 Ω. The shutoff temperature of the circuit is set by selecting the value of R80, as described under 'Alignment'.

The use of a CPU heatsink with a built-in fan instead of a plain heatsink is an unusual feature. In principle, any standard CPU cooler can be used. What is important is that the transistors are mounted insulated from the heatsink, preferably using ceramic insulation washers with heat conducting paste on both sides. The holes for the fixing screws should be drilled to 2.5 or 2.7 mm and tapped for an M3 thread. For safety, check the assembly with an ohmmeter before installing the heatsink, to be sure that the transistors are truly insulated from the heatsink. Since the transformer can deliver a very high current (not to mention a battery pack), any short circuit could have unpleasant consequences.

When mounting the boards in the enclosure, take care that the cooling air flow is not obstructed. The metal panel that fits into the rear of the enclosure should have a suitably-sized opening for the fan. Additional ventilation slots can be made in the enclosure to improve the cooling.

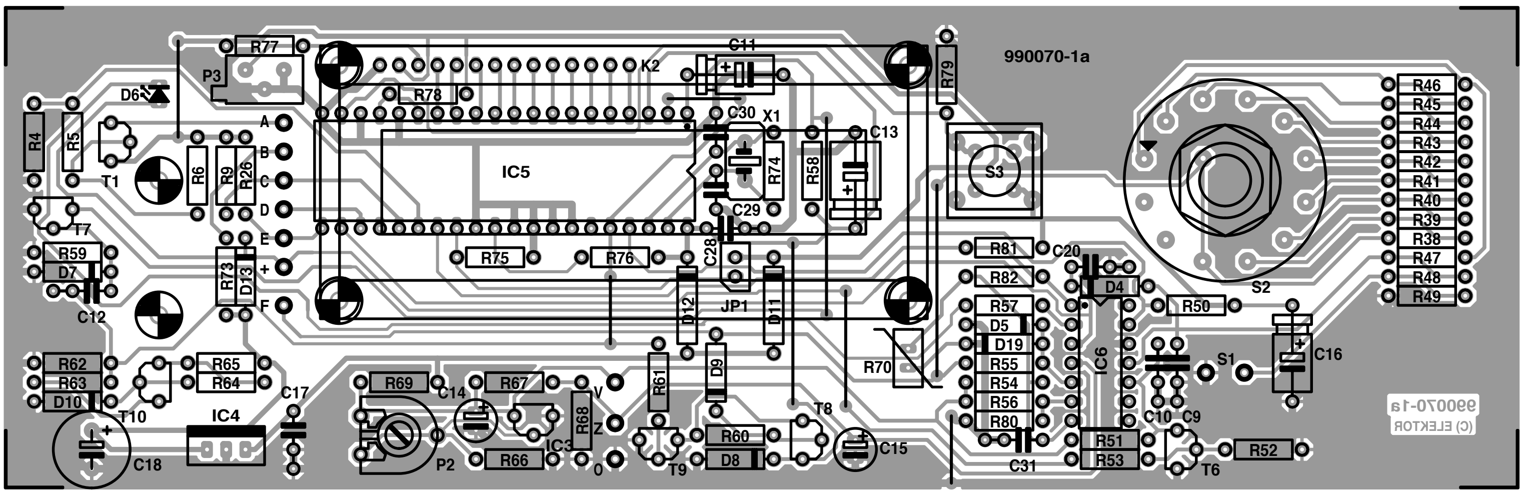
Determining the value of the series fan resistor R72 is described under 'Alignment'.

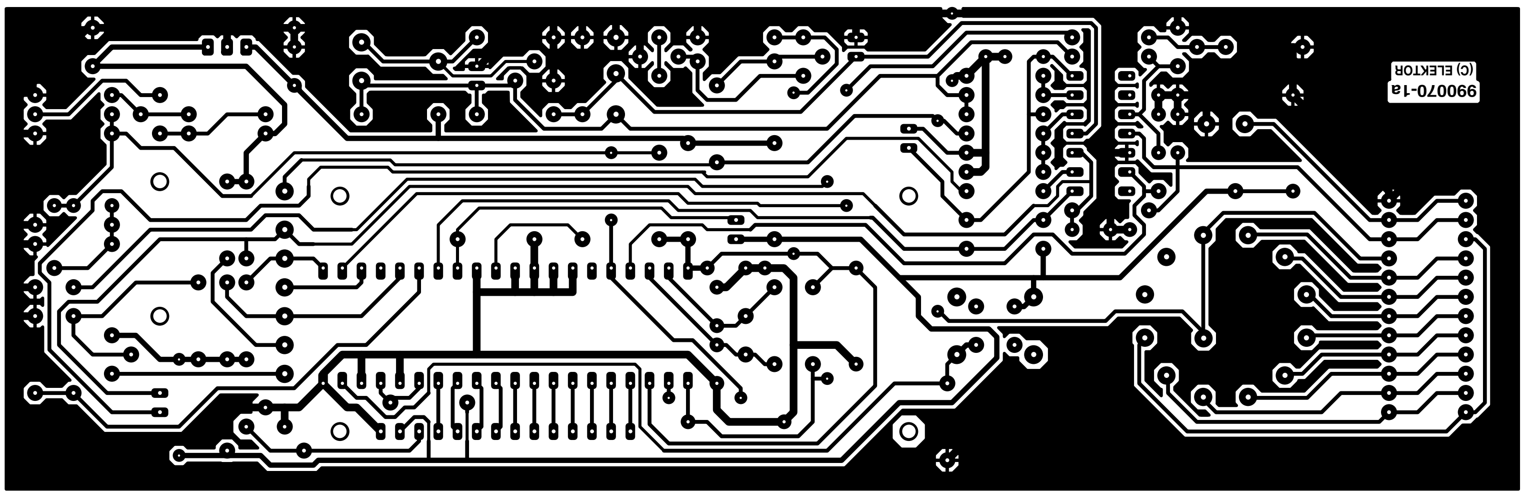
The photos show the fully-populated boards and how they are fitted into the enclosure. When connecting the transformer to the mains cable sockets, take particular care that the insulation is adequate. All components that carry mains voltage must be fully insulated against contact with the user.

The transformer usually has four secondary leads. The leads for the two 'outer' ends of the winding are grey and red, while the yellow and blue leads can be connected together to form the centre tap.

Make the ten connections between the two circuit boards using flexible low-duty wire, but use heavier-duty wire for the earth connection (around the same diameter as used for the mains connection). In the bottom of the enclosure, drill suitable holes for attaching the smaller circuit board and fixing the ring-core transformer.

Figure 4. The larger circuit board with the microcontroller, controls and display is located behind the front panel.





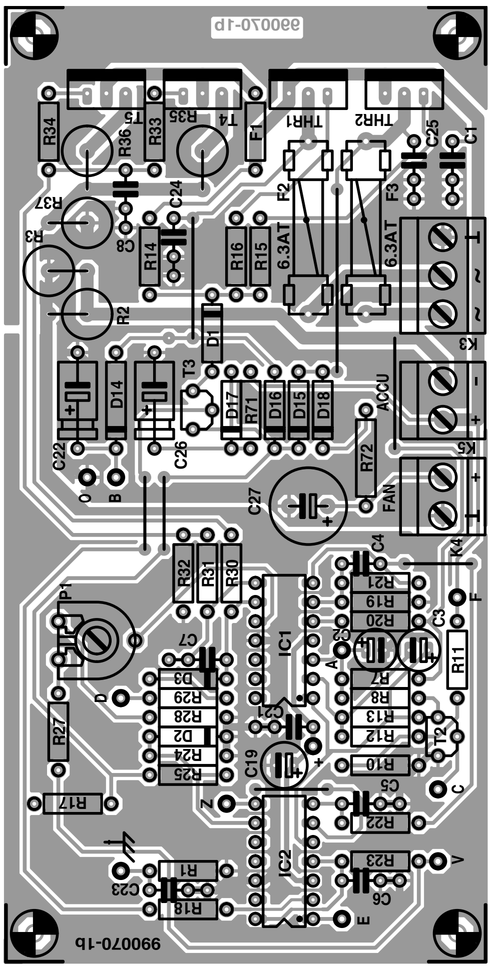
COMPONENTS LIST

Resistors:
R1,R25,R67,R75,R76,R82 = 100kΩ
R2,R3,R37 = 0Ω1 5W
R4,R21,R52,R79,R81 = 22kΩ
R5,R9,R14,R18,R61,R66,R71 = 10kΩ
R6,R78 = 47kΩ
R7 = 33kΩ2 1%
R8 = 15kΩ4 1%
R10 = 750Ω 1%
R11 = 1MΩ 1%
R12 = 9kΩ09 1%
R13 = 215kΩ 1%
R15,R16,R33,R34 = 100Ω
R17,R53,R56,R77 = 4kΩ7
R19,R20 = 3kΩ3
R22,R64 = 1kΩ
R23 = 487kΩ 1%
R24 = 33kΩ
R26,R50,R59 = 220kΩ
R27 = 3kΩ83 1%
R28 = 237Ω 1%
R29,R62 = 15kΩ
R30,R68 = 2kΩ2
R31 = 390Ω
R32 = 27kΩ
R35,R36 = 0Ω27 5W
R38-R46 = 20kΩ 1%
R47,R48 = 10kΩ 1%
R49 = 3kΩ32 1%
R51 = 178kΩ 1%
R54 = 5kΩ11 1%
R55 = 470kΩ
R57 = 5kΩ6
R58 = 150kΩ
R60 = 1MΩ
R63 = 1kΩ8
R65 = 1kΩ2
R69 = 470Ω
R70 = NTC, 500Ω *
R72 = 27Ω 2W *
R73 = 220Ω
R74 = 10MΩ
R80 = 220Ω *
R83 = VDR S10K275 (Siemens)
P1 = 100Ω preset H
P2 = 1kΩ preset H
P3 = 1kΩ preset V

Capacitors:
C1,C8,C12,C17,C20,C21,C23,C24,C25, C28,C31 = 100nF ceramic
C2,C3 = 1µF 16V tantalum bead
C4 = 1nF raster 5mm
C5 = 10nF
C6,C10 = 22nF
C7 = 22nF
C9 = 1nF
C11 = 1µF 16V
C13,C19 = 47µF 16V
C14,C15 = 10µF 63V radial
C16,C26 = 10µF 63V
C18,C27 = 220µF 35V radial
C19 = 47µF 16V radial
C22 = 22µF 35V
C29,C30 = 22pF

Semiconductors:
D1-D4,D7,D9,D11,D12,D13,D15,D16, D17,D19 = 1N4148
D5 = BAT85
D6 = LED high efficiency
D8 = zener diode 6V8, 400mW
D10 = zener diode 5V6, 400mW
D14,D18 = 1N4001
T1,T3,T10 = BC557B
T2,T9 = BC547B
T4,T5 = BUZ11
T6,T7 = BC548C
T8 = BF245B or BF 256B
THR1,THR2 = TIC116A or BT151-500R
IC1 = LM324 (DIL14)
IC2,IC6 = LM339 (DIL14)
IC3 = TL431CLP
IC4 = 7806
IC5 = 68HC05C4 (Harris), Conrad Electronics order code 692265

Miscellaneous:
JP1 = jumper
K1 = mains appliance socket with integral fuse 630 mAT and mains switch
K2 = 14-way pinheader
K3 = 3-way PCB terminal block
K4,K5 = 2-way PCB terminal block
S1 = on/off rocker switch
S2 = rotary switch, 12 contacts, 1 pole, PCB mount
S3 = pushbutton, 1 make contact
F1 = POLYFUSE 1A6 (Polyswitch, Conrad Electronics o/n 53 60 83)
Tr1 = toroidal transformer 2 x 18V 3.33A
X1 = 4MHz quartz crystal
F2,F3 = 6.3AT with PCB mount holder and cap
CPU ventilator
Case: Bopla Lab 223 mm x 72 mm x 199 mm (Conrad Electronics o/n 52 33 48) with front panels (Conrad Electronics 52 33 72)
LCD-Module 1 line, 16 characters (Sharp LM16155)
4 off TO220 mica washer with plastic bush
2 off wander socket dia. 4mm
PCB, order code **990070-1 (**see Readers Services page)
*) see text



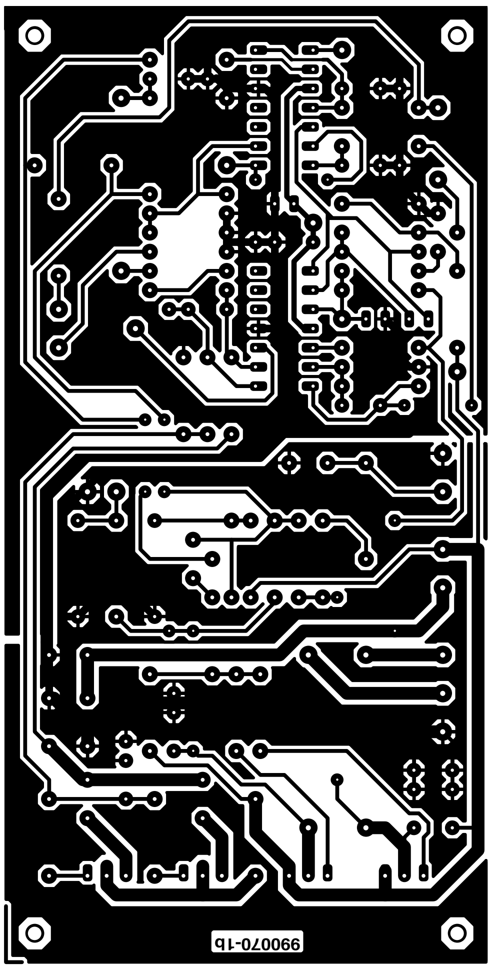


Figure 5. The smaller circuit board with the power components of the charger circuit.

Mounting the larger circuit board at the front of the enclosure is relatively simple — just slide it into the proper slot in the lower half of the enclosure.

ALIGNMENT

The first task is to match R72 to the actual CPU cooler used. The controller need not be installed for this. Connect the fan to K4 and measure the voltage across K4. If this differs from the nominal fan operating voltage (usually 12 V) by more than 1 V, the value of R72 must be appropriately modified. For testing, you can briefly use standard 1/3-W resistors if you do not have a suitable stock of 2-W or 5-W resistors. The fastest way to determine the correct resistance is to first solder in a resistor with too high a value (such as 47 Ω) and then hold a second resistor by hand 'across' this resistor (in parallel) while observing the voltage at K4. Start with a relatively high value for the second resistor, and then try successively lower values until the correct fan voltage is obtained. Next measure the effective resistance of the parallel resistors, and then solder a 2-W resistor with this value in place on the circuit board.

The controller is also not needed for the alignment of the **temperature monitoring circuitry**. First bring the unmounted NTC sensor (with a cold resistance of 500 to 1000 Ω) to the desired cutoff temperature, and measure its resistance at this temperature. Choose a resistor with the same value for R80, and mount it on the board. Mount the NTC sensor where the temperature is to be monitored, such as on the transformer or the heatsink. If you wish to monitor the battery temperature during charging, the NTC sensor must be fastened to the battery (using an elastic or sticky tape, for example), and the cutoff temperature should be around 45°C. If the transformer or heatsink is monitored, the cutoff temperature can be 60° to 90° C.

For the remaining alignments, all components must be installed, including the controller and the LCD module. To start the microcontroller-controlled alignment procedures, put S1 in the 'NiCd' position (closed), set S2 to position '6', install jumper JP1 and switch on the charge/refresh unit. Now adjust P3 (display contrast) for the best legibility. The display will read 'START SELFTEST'. Remove JP1.

A multimeter with a current range of 3 to 10 A d.c. is needed for the following alignments. Connect it between the positive and negative terminals of K5, where it will measure the output current of the battery charger flowing through the meter. Since the current is pulsed, true-RMS meters will give somewhat different readings than ordinary meters, but the differences are slight. Adjust the charge and discharge currents as follows:

1. Press S3. The display will read 'CHARGE = 3 A MAX'. Adjust the indicated value of the current to 3 A using P2.
2. Press S3 again. The display will read

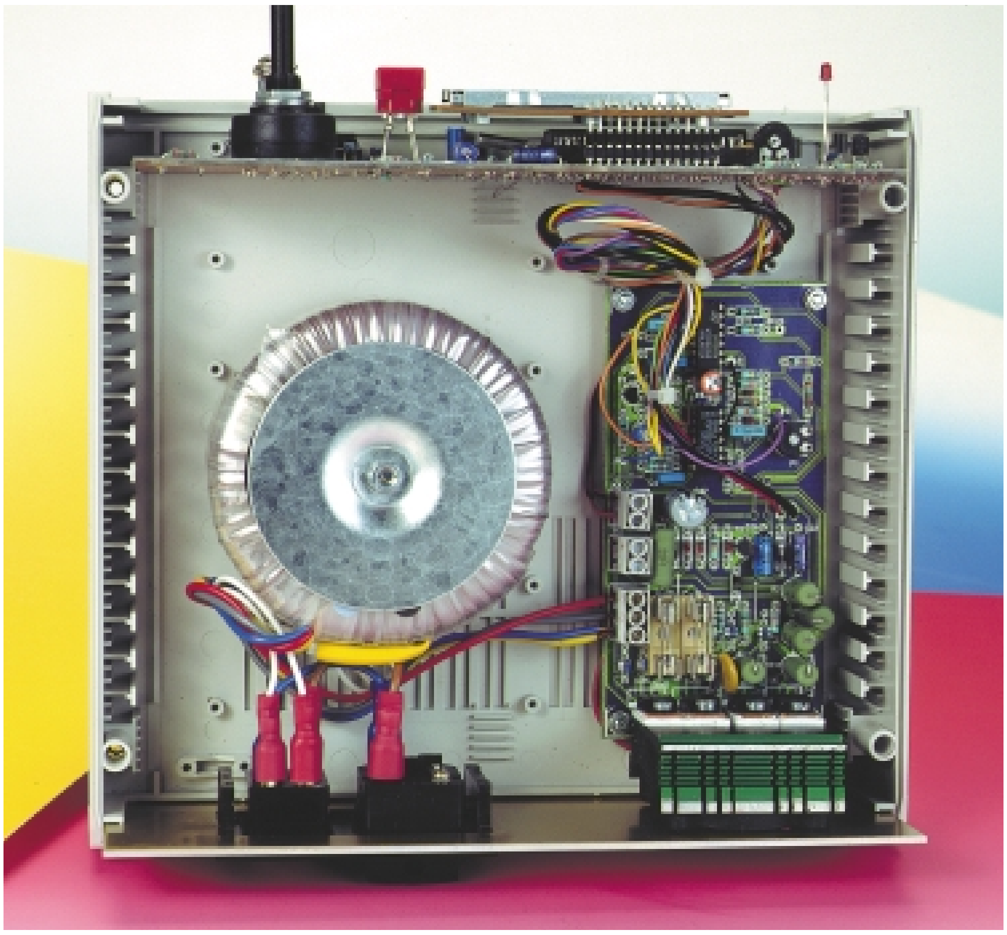


Figure 6. Top view of the prototype.

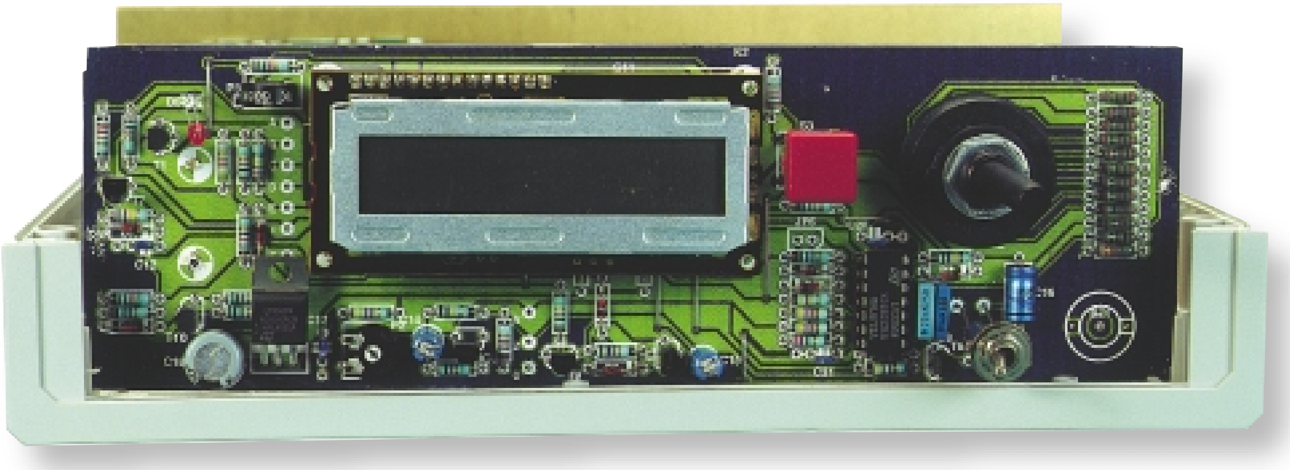


Figure 7. Front view of the prototype.

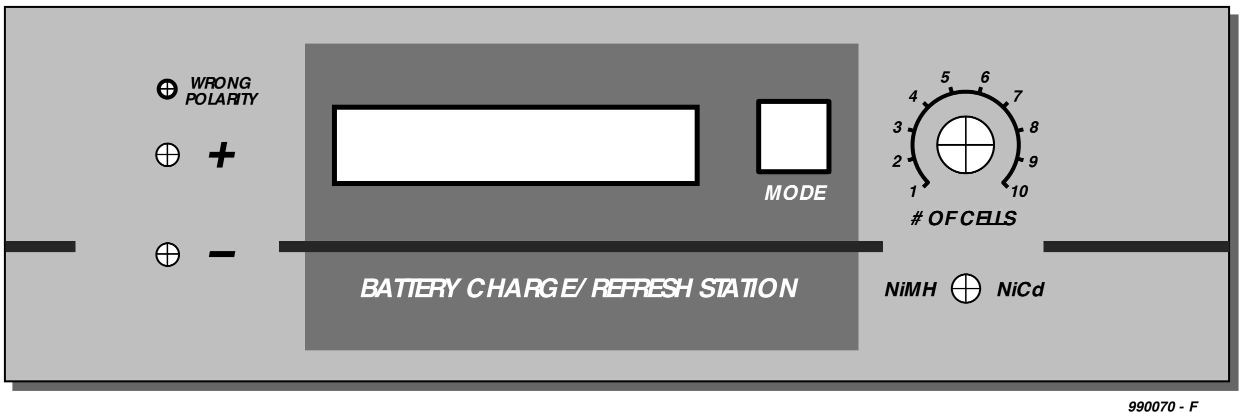


Figure 8. Suggested front panel layout (reproduced at 75% of true size).

'CHARGE = 2 A MID'. The indicated value should remain at 3 A however.

3. Press S3 again. The display will read 'CHARGE = 1 A MIN'. The indicated value should be within 10% of 1 A. Disconnect the meter.
4. Press S3 again. The display will read 'ADJUST 1.800 A'. Connect a **fully-charged** battery containing 4 to 8 cells together with an ammeter in series with the battery (between the positive terminal of K5 and the positive terminal of the battery). Adjust P1 for a charging current between 1.78 and 1.82 A. Disconnect the meter and battery.
5. Press S3 again. The display will read 'OVER-VOLTAGE'. Set S2 to position '**8**' (8 cells). The display will read 'IN:xxxx EMP:1220'. The value 'xxxx' must be greater than 1200. Next put S1 in the 'NiMH' position (open). Now either the value of 'xxxx' must be greater than 1800, or the display must read 'OVER-VOLTAGE' (the displayed values depend on the no-load voltage of the transformer).
6. Next, short the positive and negative terminals of K5 together to produce an intentional short circuit. The displayed value must be less than 10, and the polarity error indicator (LED D6, 'wrong polarity') must light. Remove the short circuit.
7. Press S3 again. The charger will now be in the normal operating mode, and the display will read 'NO ACCU TO SERVE' ('accu' = battery), since no battery is connected.

OPERATION

Switch on the charge/refresh unit with no battery connected. It will display 'NO ACCU TO SERVE', which indicates that it is ready to use. As long as no battery is connected, pressing S3 has no effect. Now select the battery type with S1 and the number of cells with S2 (1 to 10), and then connect a battery or battery pack. As soon as the message 'ADJUST: CHARGE' appears, you have five seconds to press S3 to select a different program (see the 'Program modes' box). If S3 is not pressed within five seconds, the CHARGE program is automatically started. If on the other hand S3 is pressed, a new program is selected each time it is pressed, in the sequence CYCLE, ALIVE (= refresh), CHARGE..... If S3 is not pressed again within five seconds after the last time it was pressed, the indicated program is started and charging is activated, as indicated by the message 'START CHARGING'. After an additional 15 seconds, the loaded capacity is indicated by the message 'CCAP = xxmAh' (for example, 'CCAP = 1.8mAh', where 'CCAP' stands for 'charged capacity').

A number of displays can be brought up during the charging process by pressing the function button. First the selected program mode is displayed for about two seconds ('CHARGE MODE', 'CYCLE MODE' or 'ALIVE MODE'), following which the discharge capacity DCAP is displayed for about five seconds if charging is taking place (for example, 'DCAP = 0.0mAh'), and then the charged capacity CCAP is displayed. For discharging, the display sequence after the push-button is pressed is different; after the mode display, CCAP is displayed followed by DCAP. If no measurement data are available, such as for DCAP when the battery is being charged in the CHARGE program, the value '0.0' is displayed.

The charging process ends when the battery no longer accepts any charge. In the CYCLE and ALIVE programs, the discharge process starts at this time, as indicated by the message 'START DISCHARGE'; otherwise the message 'CHARGER FINISHED' indicates that the battery may be discon-

Capacities

The measure of the status of a battery is its capacity, which means the total current that can be drawn from it, expressed in ampère-hours (Ah) or milliampère-hours (mAh). There is a relatively strong dependence between the capacity and the level of the discharge current. The lower the discharge current, the lower the losses during discharging, and thus the higher the capacity. When buying batteries, you should pay attention to the discharge current at which the capacity is specified by the manufacturer. In this regard, the concept 'C-rate' is often used, in which the current level (in A or mA) is specified as a fraction of the rated capacity (in Ah or mAh). For example, a capacity of 1 Ah at C/10 (0.1 C) means that the capacity was measured at a discharge current of 100 mA. If a different manufacturer specifies the same capacity at a higher C rate (such as C/3 = 0.33 C), the latter battery is in principle better. In this example, if a capacity of 1 Ah is achieved at a discharge current of 333 mA, you can rest assured that the capacity at 100 mA will be higher.
You should not overlook the fact that this high-performance charger works with a high discharge current (initially 1.5 A, dropping to 0.5 A). The measured capacity under these conditions applies to high-current loading of the battery and will thus always be somewhat lower than the value specified by the manufacturer, except for large and/or very high-performance batteries.
The amount of current accepted by the battery while it is being charged does not give any particular indication of the capacity of the battery. It is always higher than the battery capacity, since part of the charging current is lost in the form of heat, increasingly as the charging process nears completion.

nected. The CCAP and DCAP values can also be called up for display by pressing the function button after charging is completed, just as during charging or discharging. Press the pushbutton to cycle through the displays.

If the charging process cannot be completed successfully, due to a defective or unsuitable battery, the message 'END WITH ERROR' will appear.

If the battery is not disconnected within an hour after the message 'CHARGER FINISHED' appears, a new maintenance charging process is automatically started, as indicated by the message 'START TRICKLE'. The display also shows the value of the maintenance-charging capacity TCAP (trickle charge capacity) instead of CCAP during this process. Since the value of TCAP is not stored, the value of CCAP determined during the prior charging process can be called up by pressing the pushbutton.

If the overtemperature function is triggered during the charging process (via the NTC sensor), the program is suspended and the display reads 'OVER-TEMPERATURE'. The program resumes after about 15 minutes, with the displayed message 'CONTINUE PROGRAM'.

If the program is interrupted due to a mains power failure or switching off the charger, all settings and values are saved. Program execution will continue correctly once mains power has been restored, with the message 'CONTINUE PROGRAM' displayed. However, this is only true if the connected battery contains at least four cells and the power outage lasts at least 20 seconds. If the battery contains fewer than four cells, the charger will respond to the power interruption with the message 'ADJUST. CHARGE', and the program will have to be manually restarted.

OPERATING TIPS

If the red LED (D6, 'wrong polarity') lights up when the battery is connected, the battery has either been connected with the incorrect polarity or it is deeply discharged. With incorrect polarity the message 'NO ACCU TO SERVE' remains visible, and you must reverse the battery connections. If on the other hand the message 'ADJUST. CHARGE' is displayed, you can proceed with the charging process just as with a battery that is not deeply discharged (ie, for which the red LED does not light).

If the red LED starts to blink when the charging process begins ('START CHARGING'), there is a protection diode built into the battery or battery pack. The battery must be connected directly to the charger (without the diode).

Program modes

CHARGE
The battery is charged once using a standard fast charge process.

CYCLE
The battery is charged, discharged and then recharged.

ALIVE
The battery is charged, discharged, recharged and again discharged. If the capacity measured during the second discharge is greater than that for the first discharge, the charge/discharge cycle is repeated. This process ends with a final charge process after at most six such cycles, or when there is no difference between successive capacity values.

If the battery has built-in overtemperature protection in the form of a bimetallic contact, the charge current will be interrupted if the overtemperature protection is activated. The charger will then display 'NO ACCU TO SERVE'. If you do nothing in this situation, the contact will close once the battery has cooled down, and the charger will continue with the charging process. In spite of this interruption, the battery will be fully loaded in the CHARGE program, even if the bimetallic contact is activated multiple times. However, the CCAP display will be reset each time charging is interrupted, so that the value displayed at the end of the process will show only the loaded capacity since the last interruption.

The settings for the number of cells and the battery type must not be changed during operation! These settings may fundamentally only be changed when no battery is connected to the charger. If the charging process is unintentionally started with incorrect battery type or cell number settings, the battery must be **immediately disconnected**, and it may be reconnected only after the settings have been corrected.

Due to the high charging currents, normal plastic battery holders are not at all suitable — the typical spiral spring contact wire will quickly become red hot. Individual cells can only be charged in special battery holders designed for high currents (such as Conrad Electronics part number 51 28 77-01). With battery packs (10 cells maximum, minimum cell size AA), these contact problems do not exist, but you should be sure to use short connecting leads with adequate cross-sectional area (at least 1 mm^2).

Charging is best performed at normal room temperature (20° C), in which case the maximum battery temperature will be 45° C. High and low temperatures should be avoided; the absolute limits are 0° C and 40° C. Cold batteries must be warmed up before being charged, and you should also let the charger warm up before use if it has been stored at a low temperature.

A bit of caution is necessary when charging the smallest-sized cell (AA or mignon). **In general, the nominal capacity must be greater than 700 mAh**. To be on the safe side, you should check the battery temperature during the first several minutes when charging these small batteries for the first time. If the (discharged) battery quickly becomes hot at the start of the process, you should stop charging, since it is evidently not suited to fast charging due to excessive internal resistance. Such a battery should ideally be discarded right away, as should a battery which the charger rejects with the 'END WITH ERROR' message. However, please remember that cadmium is a highly poisonous substance, so that such batteries should be properly disposed of in an environmentally safe and responsible manner (contact your local council for information on battery disposal).

(990070-2)

fast internet access by ADSL technology

via standard (copper) telephone lines

Since the early 1990s, it has been possible to interconnect a television receiver with the telephone system to provide a user interactive data service. A few years later, it was found that when Asymmetrical Digital Subscriber Line (ADSL) technology is used, a Video On Demand (VOD) service can be implemented. Since then researchers have discovered that ADSL provides an opportunity of significantly increasing the speed of access to the Internet. This article describes the basics of the ADSL technology and how it may be used to access the Internet at high speed.

By G. Kleine

INTRODUCTION

Unshielded twisted-pair copper wires as used in telephone networks to carry voice signals in the 300–3400 Hz band are capable of transporting much higher frequencies. This capability has already been used for some time in Local Area Networks (LANs) at data rates exceeding 10 MHz.

The line attenuation up to about 6 MHz is of the order of 0.7 dB/kHz with almost constant group delay. Because of this, there is very little pulse distortion to a digital signal and, therefore, very few bit errors.

These characteristics allow the frequency spectrum above the voice band to be used for a low bit rate (up to 64 kbit/s) for an upstream subscriber's control link and a downstream digital data service at rates exceeding 6 Mbit/s. Filters with high stop-band attenuation at each end of an ADSL network are, of course, required to use both bands simultaneously.

So as to limit the bandwidth needed, modern modulation techniques such as Carrierless Amplitude Phase (CAP) modulation as used for Group 2 fax machines, and Discrete Multi-Tone (DMT) modulation, are used. These techniques allow several bits to be represented by one transmission symbol. ADSL is named for this asymmetric bit rate allocation.

In Carrierless Amplitude Phase modulation, the bit stream is first split into two components and then separately passed through non-recursive digital filters that have an impulse response differing in phase by $\pi/2$. The outputs are then added, passed through a digital-to-analogue converter (DAC), and filtered before being passed to the transmission network.

Discrete Multi-Tone modulation is very similar to Coded Orthogonal Frequency Division Multiplex (COFDM) since the main channel is split into many sub-channels.

Each serial input signal is first encoded into parallel format and then passed through a Fast Fourier Transform (FFT) processor to convert the frequency-domain samples into time-domain values with a sliding time-window effect. These values are transcoded into a serial format and passed through a digital-to-analogue converter (DAC) before transmission.

The ADSL technology has been laid down in ANSI (American National Standards Institute) Standard T1.413.(1997).

FREQUENCY SPECTRUM

Currently, the most widely modulation used is DMT. As is to be expected, an ADSL-DMT signal consists of a great number of time-domain sub-channels that are superimposed on to the twisted-pair copper telephone wires. A diagrammatic representation of the resulting spectrum is shown in **Figure 1a**. The ADSL Standard provides for the frequency range of 0–26 kHz to be left free for the analogue telephone service (POTS= Plain Old Telephone Service – colloquial). The 26–1130 kHz band accommodates 256 sub-channels each 4.3125 kHz wide. The centres of these sub-channels are also separated by 4.3125 kHz.

The individual carriers in the downstream and upstream ranges are quadrature amplitude modulated[1)] and carry between 2 bit/s/Hz and 15 bit/s/Hz. The allocation of this rate of information is adaptive, that is, during the initialization process the individual carriers are allocated various signal spaces, depending on the noise in the relevant channel: (128-QAM, 64-QAM, 32-QAM, 16-QAM, 8-QAM, QPSK). The larger the signal-to-noise ratio, the larger the signal space and thus the number of bits representing a transmitted symbol.

Clearly, each channel in the signal can transmit up to 64.7 kbit/s, which, in theory, gives a maximum capacity of more than 16 Mbit/s in the case of 256 channels. However, in practice, owing to attainable signal-to-noise ratios, only about half of this capacity can be used.

When noise levels are high, or connection cables are very long, the signal space is reduced to an extent where secure communication is maintainable. This means that there may be channels which, owing to prevailing noise or high attenuation, are useless.

The standard foresees two possible means for allocating the channels to the downstream or upstream band: relatively straightforward Frequency

1

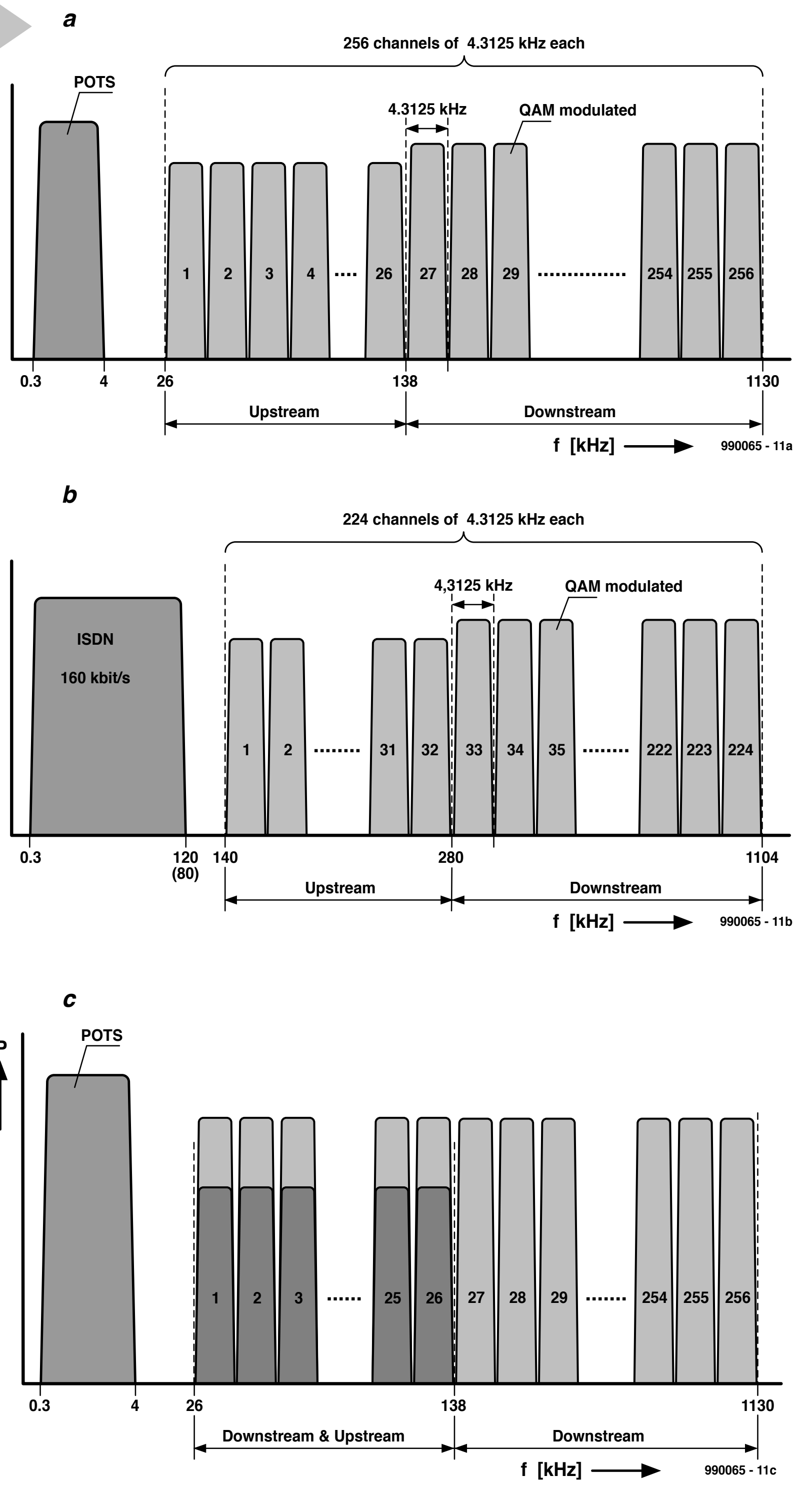


Figure 1. Spectrums with the use of ADSL technology:
(a) analogue telephony (frequency division multiplex—FDM)
(b) ISDN telephony (frequency division multiplex—FDM)
(c) analogue telephony (operation with echo compensation)

2

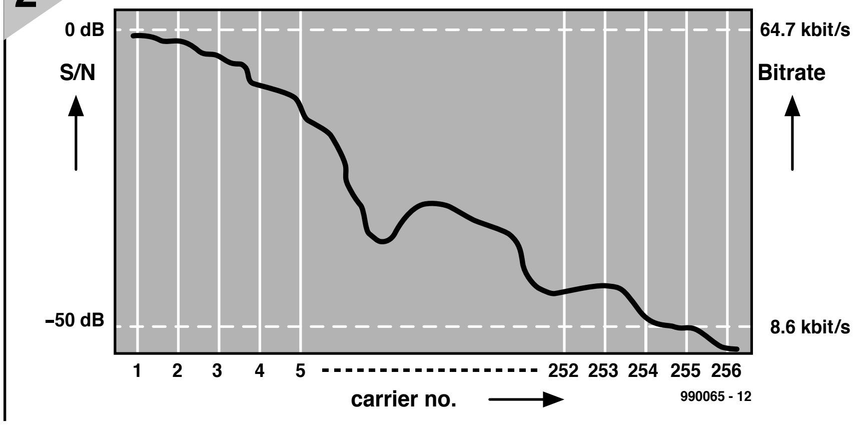


Figure 2. Bit distribution as a function of the signal-to-noise ratio.

Division Multiplexing (FDM) or Echo Compensation.

In FDM, the frequency range is split into two bands. The first 26 channels form the upstream band, while channels 27–256 carry the downstream data—see **Figures 1a** and **1b**.

In Echo Compensation, the frequency range is split into downstream and upstream according to the direction of transmission—see **Figure 1c**. This leads to a higher capacity for downstream data flow, since the lower 112 kHz of the ADSL range contains the better channels: at higher frequencies, the attenuation rises. So as to ensure that all functions well, an Echo Equalizer is needed, spaced well away from the remaining data flow. The ADSL Standard terms this mode of operation *Category 2 ADSL.*

Level differences may occur between the low frequency carriers and the high-frequency ones. If these do not exceed 50 dB, they are compensated by the channel equalizer in the ADSL modem. Higher attenuation would make the carrier ineffectual, were it not for the use of alternating line coding (in accordance with the prevailing signal-to-noise ratio) in conjunction with the channel equalizer in the modem. **Figure 2** shows a typical bit distribution as a function of the signal-to-noise ratio.

ADSL AND ISDN

Figure 1b shows the situation when an ISDN (Integrated Services Digital Network) access line is available. The signal on such a line (2×64 kbit/s) extends in some countries up to 80 kHz and in others up to 120 kHz. In order to make use of the ADSL technology, a way must be found to combine it with ISDN technology. This could, of course, be done by a switch to select between the two different usages, but this would mean that ADSL and ISDN would no longer be independent of one another and could not be used simultaneously. The solution adopted is for the DMT signal to start at 140 kHz instead of at 26 kHz. This means that, given a channel spacing of 4.3125 kHz, there are only 224 channels available. This is laid down as *Annex B* of the ADSL Standard.

The problem is caused by the fact that, in line with the ADSL Standard, the lower channels are used during the link set-up for testing the lines by means of test data packets, and for determining the bit rate of each individual channel. Since, in the case of ISDN, these channels are no longer available, channels in the upstream range are allocated for test purposes. All this is detailed in Annex B.

Since the lower channels are used for link set-up testing, they are nevertheless used in the domestic subscriber's system. They also ensure a secure first link between the ADSL modem and the end of the (telephone) trunk system. In the United Kingdom, most subscribers are within about 3 km (about 2 miles) of the end of the trunk system.

ADSL MODEM DESIGN

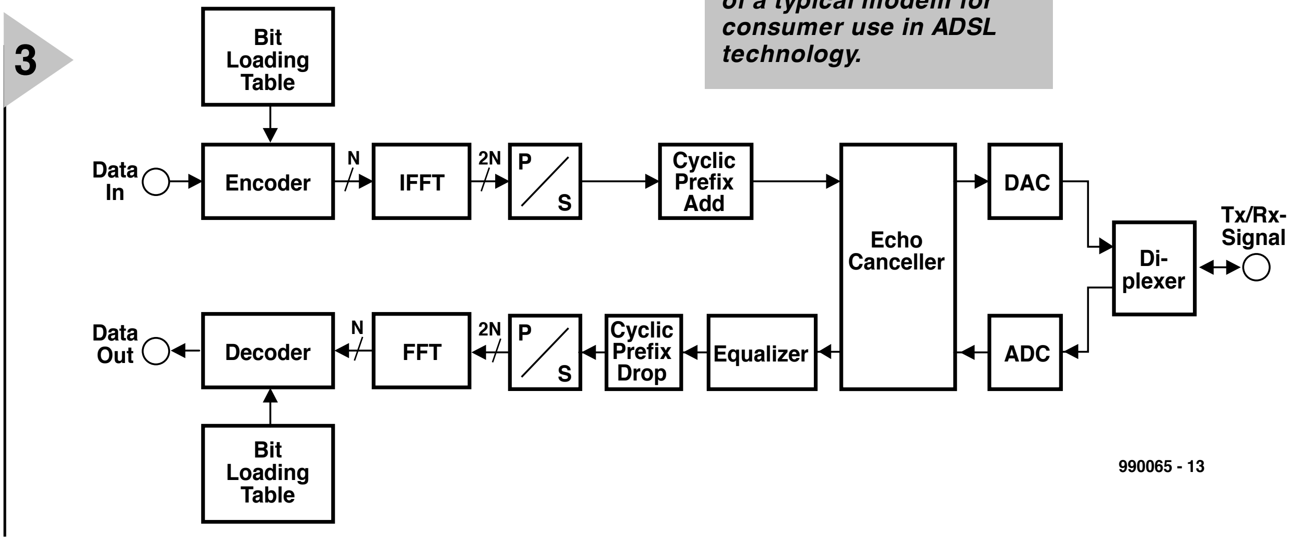
The design of a modem for use with ADSL is typically as shown in the simplified block diagram in **Figure 3.** A similar modem is also normally available at the telephone exchange, but there it can usually handle a number of subscribers and is called Digital Subscriber Line Access Multiplexer (DSLAM).

The incoming data (at the subscriber's end, the upstream data, and at the exchange end, the downstream data) are applied to an encoder, which allocates them to the *n* channels of the DMT signal. This is done in accordance with a bit loading table that has been established during the link set-up. This table shows how many bit/s each channel can handle. The encoder also provides Forward Error Control[2)] (FEC) with a Reed Solomon code (as used in digital television).

The parallel digital bit stream from the encoder is applied to an Inverse Fast Fourier Transform (IFFT) processor. This converts the *n*-bit wide frequency-domain samples into time-domain values (2*n* bits – real and imag-

Figure 3. Block diagram of a typical modem for consumer use in ADSL technology.

3



inary parts). These values are transcoded into a serial format, whereupon a cyclic prefix is added.

The echo canceller creates an image of the send and receive signals which blots up the real echoes when they come along. It is set up during the onset of the link with the aid of a test bit packet.

The output of the echo canceller is applied via a digital-to-analogue converter (DAC) and diplexer to the telephone line.

The incoming signal is applied via the diplexer to an analogue-to-digital converter (ADC) which converts the analogue signals on the telephone line into a digital data stream.

Subsequently, the signal is applied to the echo canceller which performs the same function as in the case of the outgoing data stream.

The equalizer, which is set up not only during the link set-up, but also during normal operation by means of test messages, provides requisite frequency equalization.

After the cyclic prefix has been removed, and the signal has been transcoded into parallel format, it is applied to the Fast Fourier Transform (FFT) processor. This stage reconverts the *n*-bit wide time-domain samples into frequency-domain values.

The decoder reconstructs the bits contained in the single DMT channels into the correct sequence by means of the Bit Loading Table with which it is programmed. The decoder also provides FEC with a Reed Solomon code which ensures that any bit errors are corrected.

Integrated circuit sets and other components for building an ADSL modem are available from many electronics retailers and mail order firms. Manufacturers of these parts are Motorola, STMicroelectronics, Alcatel, Broadcom, Globespan, and Texas Instruments. The web addresses of several of these manufacturers are given in **Table 1**.

The building of the modem presents constructors with a few challenges. For instance, the DMT signal needs very high gain amplifiers and linear operation of the power drivers. The crest factor is very high, which requires lots of reserve power in driver circuits. However, manufacturers like Burr-Brown and Analog Devices have special ICs available for these purposes. There are also problems in telephone exchanges, since each ADSL connection to a subscriber requires a power of 12 W. When many modems are contained, heat dissipation may become a problem.

ADSL EQUIPMENT

Figure 4 shows in diagrammatic form what an ADSL link entails at the sub-

Some abbreviations and acronyms:

AAL	*ATM Adaptation Layer*
ADC	*Analogue-to-Digital Converter*
ADSL	*Assymmetrical Digital Subscriber's Line*
ANSI	*American National Standards Institute*
ATM	*Asynchronous Transfer Mode*
ATU	*ADSL Transceiver Unit*
B-ISDN	*Broadband ISDN*
CAP	*Carrierless Amplitude/Phase modulation*
CDSL	*Consumer Digital Subscriber Line*
CODEC	*COder-DECoder*
COFDM	*Coded Orthogonal Frequency Division Multiplex*
CPE	*Customer (subscriber) Premises Equipment*
CRC	*Cyclic Redundancy Check*
DAC	*Digital-to-Analogue Decoder*
DMT	*Discrete Multi-Tone modulation*
DSL	*Digital Subscriber Line*
DSLAM	*Digital Subscriber Line Access Multiplier*
DTE	*Data Terminal Equipment*
DTMF	*Dual Tone Multi Frequency*
ETSI	*European Telecommunications Standards Institute*
FDM	*Frequency Division Multiplexing*
FEC	*Forward Error Control (or Correction)*
FFT	*Fast Fourier Transform*
IDSL	*ISDN Digital Subscriber Line*
IETF	*Internet Engineering Task Force*
IFFT	*Inverse Fast Fourier Transform*
IP	*Internet Protocol*
ISDN	*Integrated Services Digital Network*
ISO	*International Standardization Organization*
ITU	*International Telecommunications Union*
LAN	*Local Area Network*
MODEM	*MOdulator-DEModulator*
MPEG	*Motion Picture Expert Group*
N-ISDN	*Narrowband ISDN*
NT	*Network Terminator*
OSI	*Open Systems Interconnection*
PABX	*Public Access Branch Exchange*
PCM	*Pulse Code Modulation*
PDU	*Protocol Data Unit*
POT(S)	*colloquial term for Plain Old Telephone System (or Service)*
PSTN	*Public Switched Telephone Network*
QAM	*Quadrature Amplitude Modulation*
QPSK	*Quadrature Phase Shift Keying*
SLIC	*Subscriber Line Interface*
TCP	*Transmission Control Protocol*
TDM	*Time Division Multiplexing*
UART	*Universal Asynchronous Receiver/Transmitter*
URL	*Uniform Resource Locator*
USB	*Universal Serial Bus*
UTP	*Unshielded Twisted Pair*
VADSL	*Very high rate ADSL*
VDT	*Video Dial Tone: an alternative term to describe ADSL*
VOD	*Video On Demand*
WAN	*Wide Area Network*

Some relevant Internet URLs:

ADSL Forum home page	*www.adsl.com*
Alcatel	*www.usa.alcatel.com*
ANSI home page	*www.ansi.org*
ATM Forum home page	*www.atmforum.org*
Broadcom	*www.broadcom.com*
ETSI home page	*www.etsi.fr*
Frame Relay Forum home page	*www.frforum.com*
GlobeSpan	*www.globespan.net*
Internet Engineering Task Force	*www.ietf.org*
ITU home page	*www.itu.int*
Motorola	*www.mot-sps.com*
St Microelectronics	*www.st.com*
Texas Instrument	*www.ti.com.sc*
Universal Serial Bus home page	*www.usb.org*

4

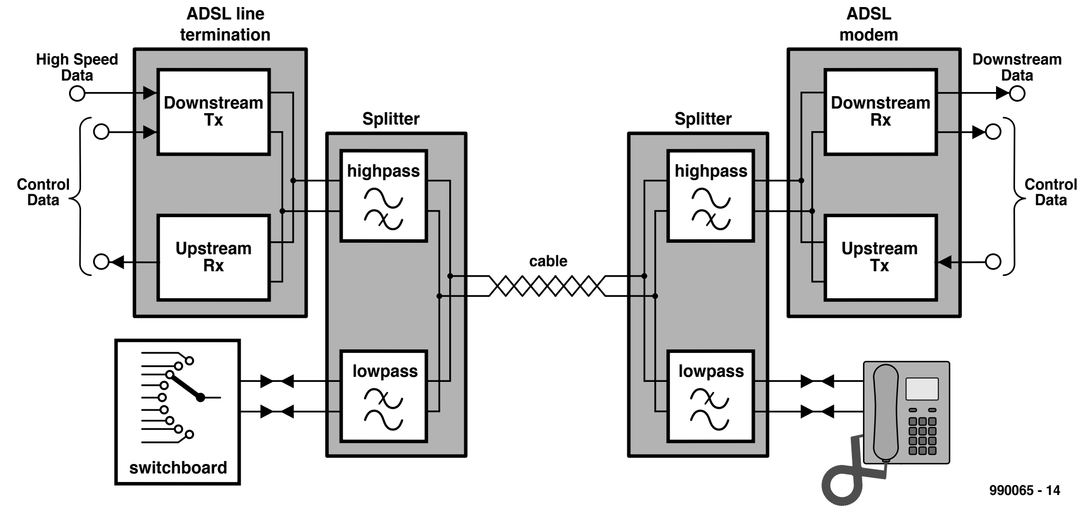


Figure 4. Simplified diagram of a typical ADSL system.

scriber's end and at the telephone exchange.

Immediately on entering the subscriber's premises and the telephone exchange, an ADSL splitter is needed. This contains a high-quality, high-pass filter with very steep skirts for the ADSL frequency spectrum.

The incoming or outgoing analogue telephone or ISDN signal, as the case may be, is passed through a low-pass filter.

At the subscriber's end, an ADSL modem must, of course, be available to which the output of the splitter is applied. This modem contains a receiver (Rx) for the high-rate downstream signal and a transmitter (Tx) for its own upstream signal. The upstream and downstream signals contain not only message data, but also management and control data.

At the telephone exchange, ADSL terminators must be added for each and every subscriber who wants ADSL service. These units are the opposite of a modem: a downstream sender (Tx) transmits the high-quality digital data stream via the splitter to the telephone lines. The upstream receiver (Rx) processes data at moderate rates. A Digital Subscriber Line Access Multiplexer (DSLAM) makes ADSL channels available to a number of subscribers.

ADSL SYSTEM STRUCTURE

Figure 5 shows in diagrammatic form what equipment other than an ADSL splitter and a modem are required. At the centre are the ADSL splitters in the telephone exchange and at the subscriber's premises.

At the exchange end, the output from the splitter is linked to the ADSL Termination and from there to the ATM (Asynchronous Transmission Mode) Backbone[3] via the ATM switch. The transmission rate between the line termination and ATM switch is 155 Mbit/s.

At the subscriber's end, the output of the splitter is applied to the ADSL modem, which contains an ATM-F25.6 interface (25.6 Mbit/s) or a (slower) LAN (Local Area Network) interface Type 10BaseT. The computer must contain a corresponding ATM or LAN card to be able to work failure-free with the modem interface.

In case an ISDN line is used, a Network Terminator (NT) must be inserted between the splitter and the ISDN connection—see **Figure 5**.

CONNECTION SET-UP

Owing to the many ways an ADSL transmission system may be set up, it is essential to study the protocol, specification, and any other relevant literature before starting the practical work.

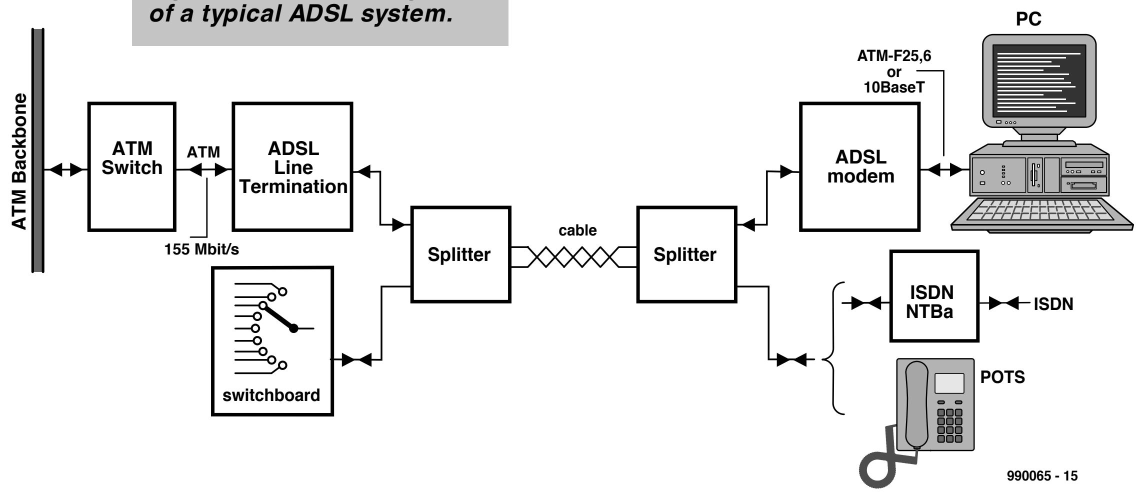
The frequency response in both directions must be measured carefully and it should be ensured that the two modems use the same carrier frequency. Subsequently, the bit rates of the upstream and downstream channels should be determined, and also which means is to be used for channel allocation (FDM or echo compensation) with the aid of test messages. It is at this point that the maximum bit rate of individual connections is decided.

ADSL technology is capable of working with varying signal-to-noise ratios. Its Bit Swapping facility allows bits to be re-allocated to a specific channel during operation.

The start phase may take from 20 seconds to one minute. This slow beginning ensures, however, that the

Figure 5. Detailed diagram of a typical ADSL system.

5



maximum possible data rates for each and every channel are fixed optimally. If, for one reason or another, it is necessary to fix the bit rates anew, it is not necessary to wait again for 20–60 seconds before operation can start. There is a short procedure for this, which takes only a few seconds. In that case, it is, however, necessary for the modem to monitor the transmission quality of each and every channel.

ADSL LITE MODEM

Shortly after the ADSL Standard had been published by ANSI in 1997, a number of manufacturers, including Microsoft, Intel, and Compaq, formed the Universal ADSL Working Group (UAWG). One of the aims of this group was to get rid of the splitter, since this would mean a substantial saving at the exchange on the Subscriber Line Interface (SLIC) and at the subscriber's end on the ATM Ethernetwork card—see **Figure 6**. It should be borne in mind that the ADSL splitter is a costly unit. The consequent G.Lite or Universal ADSL modem is standardized by the ITU in the *ITU Standard G992.2 – Splitterless ADSL.*

Apart from the introduction of this modem, the number of channels is reduced from 256 to 128, the number of bits per second per Hertz is reduced, so that the signal space is smaller, the downstream bit rate is reduced to 1.5 Mbit/s, although the upstream can still be sent at 500 kbit/s. Furthermore, the output level is reduced to such an extent that power consumption and the requisite linearity of the analogue driver stages are moderated significantly. As a bonus, this also helps to keep the analogue telephone traffic almost entirely free of interference. And last, but not least, with an ADSL Lite modem, operation is in Category 2 of the ANSI Specification, which means that the upstream and downstream sections share the lower ADSL frequency range by means of echo compensation. This guarantees good transmission coefficients on the individual carriers.

It is interesting to note that splitterless ADSL technology is much more popular in Anglo Saxon countries than in continental Europe.

CONCLUSION

Now the ADSL technology has proved itself among commercial users, it is clear that it can lead to better use of the telephone system, particularly as regards access to the Internet, by domestic subscribers Even when downstream rates of only (!) 1.5 Mbit/s are attainable, this gives a 27-fold increase in data rates compared with those provided by a 56 kbit modem.

Current modems for domestic subscriber use are of the hybrid type, which can handle the V.90 analogue standard as well as the ADSL standard. Most of these modems can be adapted via firmware* in case of standard updates.

6

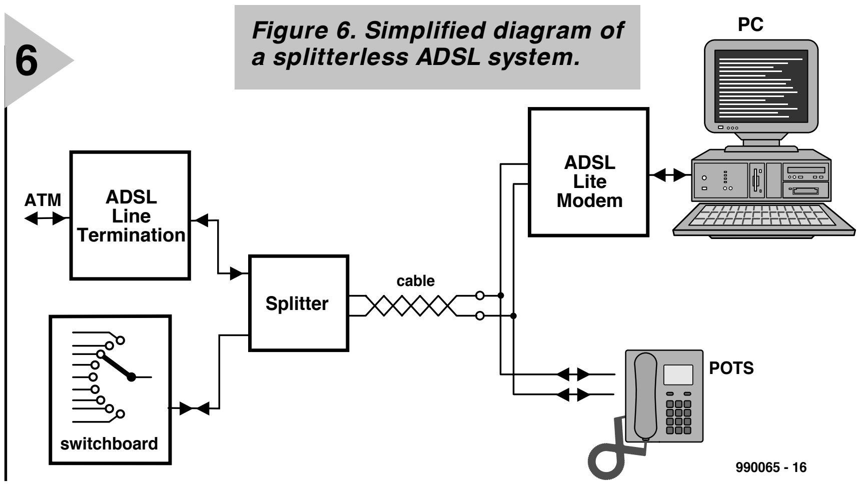


Figure 6. Simplified diagram of a splitterless ADSL system.

AND THE FUTURE?

Technology does not stand still, and already there are countries where Very high rate ADSL—VDASL—is being developed or market-tested. With the advent of fibre technology, the line impedance is being greatly reduced. This allows for very much higher bit rate services: currently expected to be 52 Mbit/s downstream and 3.2 Mbit/s upstream (with a copper tail of, perhaps, 100 metres) within a few years' time. Such rates will make MPEG-2 data transmissions possible.

[990065]

Notes:

1) Quadrature Amplitude Modulation (QAM) digital is a variant of Quadrature Phase Shift Keying (QPSK). In QPSK, quadrature phase shift of the carrier is used to convey two bits of data in the same bandwidth as one bit. In QAM digital, this is extended by obtaining 8, 16, 32, 64, 128, 256 phasors from the same carrier frequency to represent 8, 16, 32, 64, 128, 256 unique binary code patterns, each of 3, 4, 5, 6, 7, 8 bits.

2) Forward Error Control (or Correction) is a technique in which the means to detect bit errors is contained within the transmitted message stream thus allowing the receiver to correct the errors without requiring retransmission of the data.

3) It should be noted that echo cancellation is beneficial only whenever the same frequency is used for bidirectional traffic .When different subchannels are used for different directions, echo cancellation is superfluous. In that case, the UTP becomes three access points: one for speech, one for upstream data, and one for downstream information.

4) The backbone is the major transmission path of a Public Data Network (PDN).

5) Strictly speaking, firmware is system software held in read-only memory (ROM).

References:

ANSI T1.413: Network and Customer Installation Interfaces – Asymmetric Digital Subscriber Line Metallic Interface. Issue 1, 1995. Draft Issue 2, December 1998.

RFC 791: Internet Protocol

ITU G992.1 (G.dmt) Asymmetrical Digital Subscriber Line (ADSL) Transceivers.

ITU G992/2 (G.lite) Splitterless Asymmetrical Digital Subscriber Line (ADSL) Transceivers.

DSL Simulation Techniques and Standards Development for Digital Subscriber Line Systems, by Walter Y Chen, MacMillan Technical Publishing, Indianapolis, IN, 1998

ADSL and DSL Technologies, by Walter Goralski, McGraw-Hill, New York, 1998.

Integrated Services Digital Networks, by Hermann J Helgert, Addison Wesley Publishing Company, Reading, Mass. 1991.

ISDN: Concepts, Facilities, and Services, by Gary C Kessler and Peter V Southwick, McGraw-Hill, New York, 1998.

ADSL/VDSL Principles, by Dennis Rauschmayer, Macmillan Technical Publishing, Indianapolis, IN, 1999.

ADSL: Standards, Implementation and Architecture, by Charles K Summers, CRC Press, London & New York, 1999.

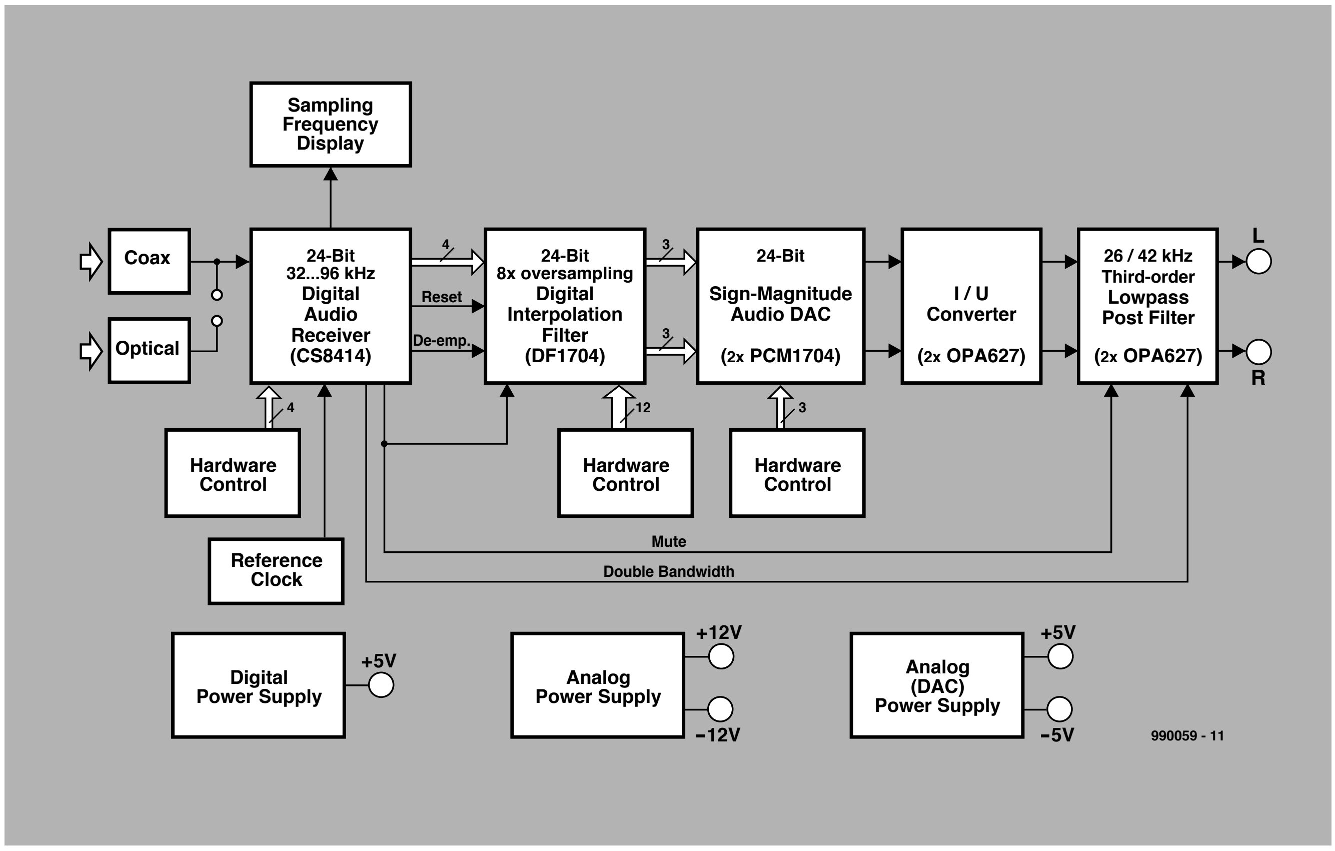
audio DAC 2000 Part 1

for perfectionists

This brand-new digital-to-analogue converter (DAC) is intended especially for those audio enthusiasts who wish to have their audio system fully up to date at the beginning of the new millennium. The 24-bit resolution and the top sampling rate of 96 kHz ensure that full advantage can be taken of the qualities of the latest compact discs (CDs) and digital video discs (DVDs).

Design by T Giesberts

1



Sampling Frequency Display
Coax
Optical
24-Bit 32...96 kHz Digital Audio Receiver (CS8414)
4
Reset
De-emp.
24-Bit 8x oversampling Digital Interpolation Filter (DF1704)
3
3
24-Bit Sign-Magnitude Audio DAC (2x PCM1704)
I / U Converter (2x OPA627)
26 / 42 kHz Third-order Lowpass Post Filter (2x OPA627)
L
R
4
Hardware Control
12
Hardware Control
3
Hardware Control
Reference Clock
Mute
Double Bandwidth
Digital Power Supply
+5V
Analog Power Supply
+12V
–12V
Analog (DAC) Power Supply
+5V
–5V
990059 - 11

Figure 1. The block diagram of the Audio DAC 2000 clearly shows the design and what are the most important parts of the unit.

INTRODUCTION

Over the past few years there have been quite a few developments in digital audio engineering. Although quality-conscious audio enthusiasts generally welcome these developments, many of them wonder if there will ever be an end to their having to update and adapt their systems. It is, therefore, perfectly understandable that the quality DACs published in this magazine over the past seven years or so have proved very popular. After all, when it comes to re-evaluating a digital audio source, it is much simpler to update just the DAC and not the entire system. Moreover, a stand-alone DAC has the advantage that it is universal and can be combined with any CD/DVD player or digital tape recorder.

The Audio DAC 2000 is intended for the new millennium: it has a 24-bit resolution and is suitable for sampling rates of 32–96 kHz. These properties make it state of the art as far as technology is concerned, while the design of the practical circuit is aimed at quality without compromise.

Some enthusiasts may wonder why higher sampling rates have not been catered for. The answer to this is that it is questionable whether higher rates will ever be implemented. Although the new 192 kHz standard is written, pundits reckon that it will take quite a few years before the hardware and software will become commercially available.

DESIGN

The circuit is contained on four individual printed-circuit boards: one for the ± 12 V and + 5 V power supplies; one for the digital audio receiver with display driver; one for a 2-digit LED display; and one for the digital/analogue circuits, the digital filter, the DACs and the analogue output stage. Its block diagram is shown in **Figure 1**.

The power supplies comprise a + 5 V section for the digital circuits (receiver and digital filter) and a ± 12 V section for the analogue output section, including the associated relays. There is also a ± 5 V supply, derived from the ± 12 V supply for the DACs.

The digital audio receiver is associated with a sampling rate display, hardware control, and reference clock.

The display consists of two 7-segment LED modules for indicating the sampling rates: 32 kHz, 44 kHz (in reality 44.1 kHz), 48 kHz, 88 kHz (in reality 88.2 kHz), or 96 kHz.

The hardware control is primarily a circuit for setting the receive mode via a 4-pole dual-in-line (DIP) switch.

The reference clock is an accurate crystal oscillator operating at 6.144 MHz which is used by a comparator in the receiver to determine the frequency of the received clock – a phase-locked loop, PLL.

The data that indicate the sampling rate, and the most important received channel-status bits (strictly speaking only the emphasis bit) are multiplexed by the receiver.

The data are demultiplexed by a Generic Array Logic – GAL™ – which also drives the display. As far as the multiplexed data is concerned, they are translated and passed to the outputs of the registers. This prevents additional switching lines coming into being: a multiplexed display would demand quite a high current.

In normal operation, the outputs of the GAL are static. A number of links needed for the display are already interconnected so as to keep the number of requisite outputs to a minimum.

The link between the digital audio receiver and the display is made by a length of 10-core flatcable, and that between the receiver and DAC board by 16-core flatcable. The 16-core cable also carries the + 5 V supply and various signals to and from the digital filter: serial audio data, power on reset, de-emphasis, mute, and switching. The

™ GAL is a trademark of National Semiconductor Corporation.
* Sony/Philips Digital Interface Format

switching signals almost double the bandwidth of the filter when sampling rates of 88.2 kHz or 96 kHz are detected.

The mute signal is actuated when there is no signal at the receiver input or when the PLL cannot lock. It is taken from the error output (pin 5 – ERF) of IC_1 and used to de-energize the output relay and to switch the digital filter to the mute mode.

The reset pulse for the receiver and digital filter is generated by network R_6-C_{13} and inverted by the GAL.

The de-emphasis signal is used by the digital filter to correct the pre-emphasis in the source signal. Twelve DIP switches determine the various settings of the filter as regards the input and output formats, the number of bits, the filter characteristic, and others.

The digital filter drives two DAC chips: one for the lefthand and one for the righthand channel. These chips can be set by hardware which will be reverted to later.

The output of each of the DACs is a pure current source. The type specified was chosen in view of its well-defined voltage, good linearity, low noise, low offset voltage and high slew rate. It is not cheap, but ideally suited to the present application.

The analogue filter at the output is needed to remove the residue of the oversampling products and the r.f. noise. It can be switched between two cut-off frequencies to allow the use of the two highest sampling rates.

Each filter section uses a double-pole relay, since a single-pole type for both channels would not give the requisite channel separation at high frequencies. This is because the *RC* sections of the filters have too high an impedance.

Since the output impedance per channel is only 100 Ω, a single double-pole relay is used at the output to switch the mute function on/off and to obviate switch-on noises.

CIRCUIT DESCRIPTION

The circuit diagram of the Audio DAC 2000 is shown in **Figure 2**.

An important task of the circuit is the decoding of the S/PDIF* data flow into a serial data format that can be used by the DACs, which is carried out by IC_1. The circuit associated with this chip is housed on a discrete board so that the coaxial and optical input connectors can be placed in the most convenient position on the enclosure.

The input impedance, which has the traditional value of 75 Ω as far as the coaxial input is concerned, is determined by resistor R_1.

The optical input is provided by IC_2, which is a standard chip used for this purpose in consumer equipment.

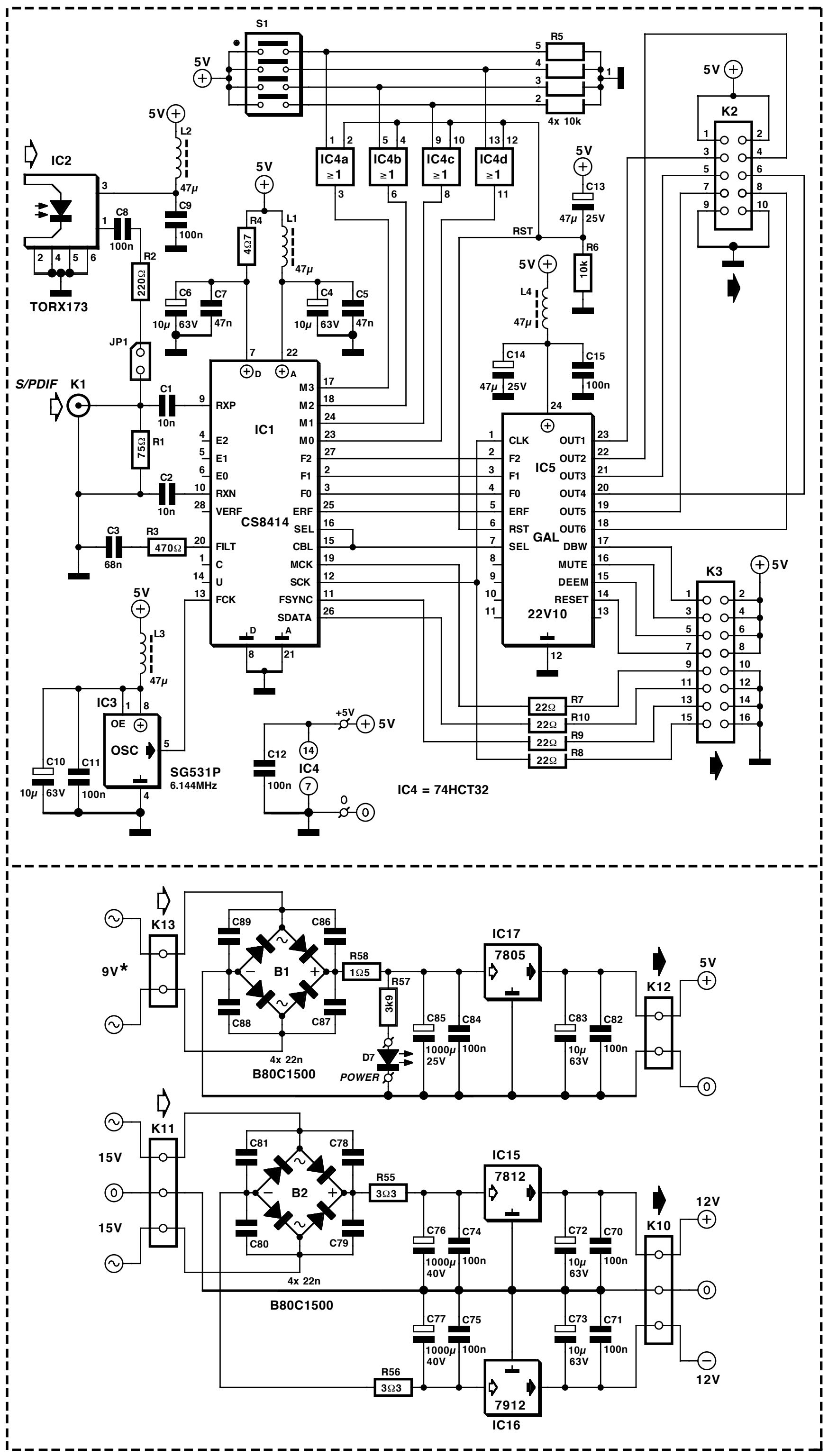
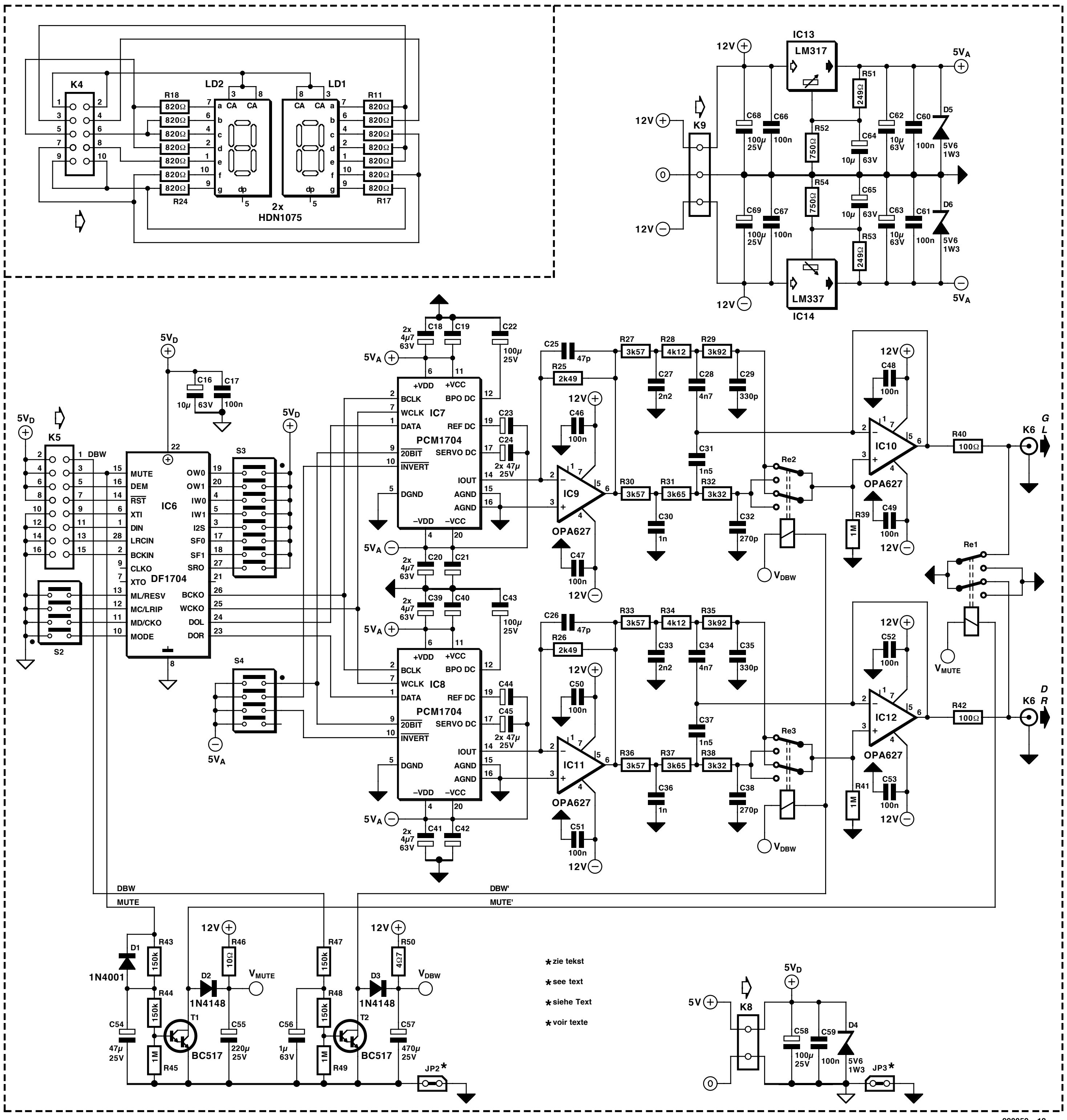


Figure 2. Circuit diagram of the Audio DAC 2000. The dashed lines show how the circuit is divided on to the four boards.

The output of the IC is applied to the input of IC_1 via potential divider R_1-R_2, whose values are chosen such that the signal across R_1 is slightly larger (0.6 V) than the standard one from the coaxial input (0.5 V). The 0.1 μF coupling capacitor prevents any d.c. on the input from reaching the receiver.

When the optical input is used, jumper JP_1 must be shorted, and the coaxial input cannot be used. It is then, however, possible to use the coaxial input as S/PDIF* output. In this case, the output impedance and signal levels are no longer standard, but the latter may be increased by slightly reducing the value of resistor R_2. In fact, the level of the input signal across R_1 may be as high as 1 V_{p-p} without any problems.

The IC uses a number of frequency sensors to ensure that the PLL locks as rapidly as possible to the incoming



data flow. In the absence of an input signal, the frequency of the voltage-controlled oscillator (VCO) is low.

The digital filter on the DAC board needs four signals, all derived from the incoming S/PDIF data by the receiver chip:

SDATA contains the serial data of both channels.

PSYNC is the lefthand/righthand clock to separate the samples for the two channels. Depending on the mode of operation, it is equal to the sampling rate, F_s, or twice that rate.

Serial clock SCK is needed for clocking the individual bits and is equal to $64F_s$.

MCK, a clock equal to $256F_s$, which is needed for oversampling and interpolation. Resistors R_7–R_{10} limit any ringing caused by the capacitive loads formed by the flatcable and the digital filter.

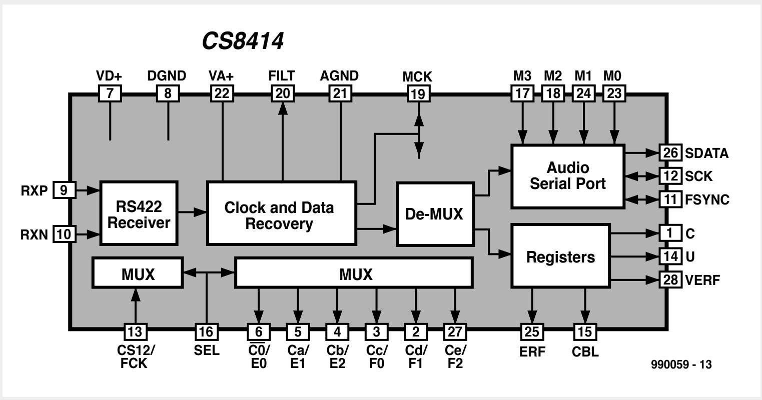
These four signals can be provided by the receiver in various standard formats: which one is determined by mode pins M_0–M_3. Details of this can be found in the CS8414 data sheets elsewhere in this issue.

The recommended mode is the I^2S mode, since in this the number of bits is basically not fixed: it may be 16-bit data or 24-bit. Because of this, the preferred setting of DIP switch S_1 is S_{1-4} ON (M_1= 1) and the remainder OFF (M_0= M_2= M_3= 0). Note that on the switch M_0–M_3 are mixed up but on the board the relevant names are adjacent to them (as is the level at ON or OFF).

Other formats are often just as they

96 kHz Digital Audio Receiver Type CS8414

The Type CS8414 is an upgraded version of Digital Audio Receiver Type CS8412, which has been used in several past articles published in this magazine. It is pin-compatible with its predecessor, but is available in a 28-pin SOIC (standard SMD) case only. A data sheet of it can be found elsewhere in this issue.



The most notable difference between the two devices is that the range of sampling rates available with the CS8414 has been extended to 96 kHz. The frequency indication with 400 ppm accuracy of the CS8412 has been sacrificed in favour of the 88.2 kHz and 96 kHz rates in the new device.

Clock detection is carried out by a 2nd-order loop filter via a phase-locked loop (PLL). In the CS8414 the design of the external RO-filter is slightly different than with the CS8412.

The input of the CS8414 is an RS422 receiver that can handle differential (symmetric) as well as asymmetric (single-ended) signals. In the Audio DAC2000 the input is configured for decoding single-ended signals only: input RXN is therefore bypassed to earth.

To ensure optimum operation of the phase detector in the internal PLL, the input is based on a 50 mV Schmitt trigger.

are named, for instance, 'MSB first right justified', whereby the location of the least significant bit (LSB) is fixed with respect to the L/R clock. The result of this is that some most significant bits (MSBs) may be missing. In the I^2S mode, the location of the MSB is fixed, so that, assuming there are more bits, only some LSBs may be 0. Some of the other formats are fully compatible with the Burr Brown digital filter, but that is left to the reader to sort out if he/she so wishes.

The various modes have been made presettable on purpose in view of future extension/expansion/upgrade or other applications. It also enables the receiver board to be used with other audio DACs. This board therefore has an extra + 5 V terminal: the 5 V line is linked to it from the DAC board via K_3.

The manufacturers recommend resetting the IC immediately after a power up, for which purpose a circuit with four OR gates contained in IC_4 is used. This circuit cannot be provided in the GAL used, since this would require four additional inputs and four outputs.

The IC is reset when all mode pins are made high, which is the reason that the DIP switch is linked to the mode input via the OR gates. The reset proper, which is also applied to the digital filter on the DAC board via the GAL, is provided by network R_6-C_{13}.

The reference frequency of 6.144 MHz which the receiver needs to determine the sampling rate is provided by crystal oscillator IC_3. The output pin of this IC is located very close to the relevant input (FCK) of IC_1 to minimize the noise level of the clock signal. The supply lines are well decoupled by network L_3-C_{10}-C_{11}.

The supply lines to all ICs are decoupled effectively: those to the analogue and digital sections of IC_1 separately.

Channel status output C, User bit output U, and Validity and ERror Flag VERF are not used.

The Channel status BLock (CBL) start is used to demultiplex the channel status output bits (pins 2–6 and 27) by linking these to select pin SEL. When SEL is low, the Error Condition (not used) and Frequency Reporting Bits are applied to the outputs, which are then called E_0–E_2 and F_0–F_2 respectively. When SEL is high, the Channel Status is applied to the relevant outputs in the shape of some status channel bits, whereupon the outputs are called C_0 and C_a–C_e. Of these, only C_c (F_0) is used, which is channel status bit C_3. In fact, this is the emphasis bit of the channel status. It is inverted by the GAL and retained via a register output, which therefore retains the actual level and does not follow SEL. The CBL output is therefore linked to an input of the GAL to demultiplex the data.

When SEL is low, bits F_0–F_2 are recoded to six register outputs to drive a 2-digit LED display. Various segments of the display are already combined so that, without the need of multiplexing, only six outputs are needed to display the five sampling frequencies on two 7-segment displays (7 mm types). The only compromise is that any value after the decimal point (rate in kHz) is omitted.

The six outputs, + 5 V, and earth, are linked to the display board via K_2, a 10-core flatcable, and a 10-pin connector. Owing to its height and location behind the front panel, a box header cannot be used on this board.

When ERF is active, all display outputs are high and only two dashes (hyphens) light. Both g-segment LEDs are permanently linked to earth via R_{17} and R_{24} respectively and therefore light permanently as long as there is supply voltage.

The SCK clock (pin 1 of the GAL) is used to clock all register outputs.

The information on the actual sampling rate is used not only for driving the frequency display, but also for switching the cut-off frequency of the analogue output filter to a higher value. That is, output DBW (double bandwidth) goes high when a rate of 88.2 kHz or 96 kHz is detected.

De-emphasis output DEEM is applied to the digital filter only. An indication of this is purposely not provided since CDs with pre-emphasis are very rare indeed. However, since the facility is there, it is used in the Audio DAC 2000 for correcting any pre-emphasis, particularly so since there is now no need for an additional *RC* network in the analogue output output filter (both channels).

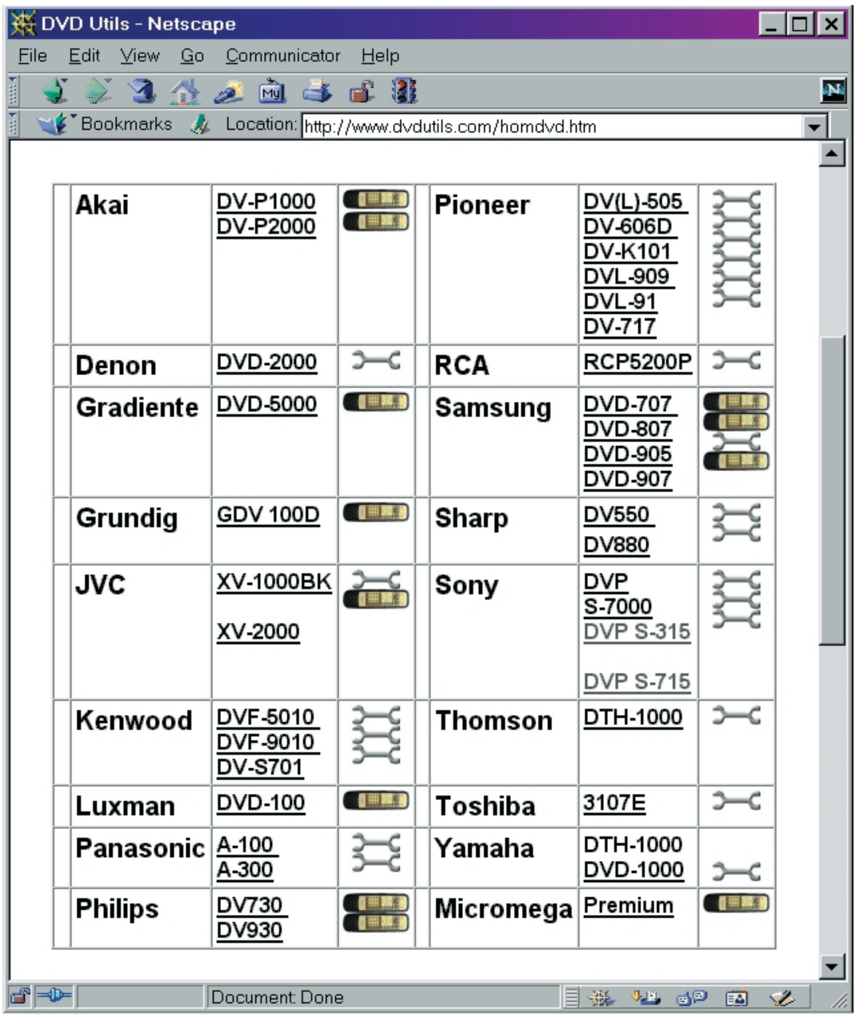
[990059]

Next month's instalment describes the DAC board with particular emphasis on the digital filter and the DACs.

modifying DVD players

how to change region codes

As a kind of sequel to last month's edition of 'electronics on-line' we have located a number of websites covering the 'hot' subject of how to modify the region code setting of consumer-market DVD players.


DVD Utils - Netscape

File Edit View Go Communicator Help

Bookmarks Location: http://www.dvdutils.com/homdvd.htm

Akai	DV-P1000 DV-P2000	▬ ▬	Pioneer	DV(L)-505 DV-606D DV-K101 DVL-909 DVL-91 DV-717	🔧 🔧 🔧 🔧 🔧 🔧
Denon	DVD-2000	🔧	RCA	RCP5200P	🔧
Gradiente	DVD-5000	▬	Samsung	DVD-707 DVD-807 DVD-905 DVD-907	▬ ▬ 🔧 ▬
Grundig	GDV 100D	▬	Sharp	DV550 DV880	🔧 🔧
JVC	XV-1000BK XV-2000	🔧 ▬	Sony	DVP S-7000 DVP S-315 DVP S-715	🔧 🔧 🔧
Kenwood	DVF-5010 DVF-9010 DV-S701	🔧 🔧 🔧	Thomson	DTH-1000	🔧
Luxman	DVD-100	▬	Toshiba	3107E	🔧
Panasonic	A-100 A-300	🔧 🔧	Yamaha	DTH-1000 DVD-1000	🔧
Philips	DV730 DV930	▬ ▬	Micromega	Premium	▬

Document: Done

The DVD (digital video disc) seems poised for a commercial breakthrough, and sales volumes of consumer DVD players rise accordingly. Unfortunately, under considerable pressure of the almighty film industry, DVD hardware manufacturers are forced to fit their products with a restriction that allows a DVD to be played in a certain region only.

This has been achieved by dividing the world into six regions. Europe and Japan, for example, represent region 2, while the U S of A are region 1. A DVD player bought in a European country will refuse to play a DVD bought in the USA and having region code 1. This is particularly bad and disappointing for film fans, because they have to wait for a film to be DVD-released in Europe, while the American version has been available for some time already.

Of course, some whiz kids came up with solutions to this problem. Most DVD players are designed and built in a way that enables the manufacturer to easily adapt each basic version for a particular region code. In most countries companies exist offering so-called 'region-free' DVD players, or an upgrade to your existing player for around 100 pounds.

The Internet is not only a vast resource for addresses of these companies, but also for do-it-yourself modification documents. Some of the most ingenious hacks consist of modifications to the firmware (i.e., the program executed by the processor inside the DVD) and tell you which bits have to be modified.

A good starting point with a generous overview is offered by **Data Testlab** at
http://www2.datatestlab.com/regionhacks/regionhacks_players.htm
On this site you will find a series of brands and types as well as links to many other sites including **Planet DVD** at
http://www.planet-dvd.ch/
CineHome at
http://www.cinehome.de/
and the French-language Dézonage at
http://perso.club-internet.fr/hitcher/dezone.html
Another site with extensive information on converting certain players goes by the name of **DVD-utils** and may be found at
http://www.dvdutils.com/homedvd.htm
Clear photographs show, for example, where to remove resistors from circuit boards, or connect extra cables.

There's even a special website dedicated to converting Sony players only: the **Sony DVD codefree** site at
http://members.xoom.com/sonydvd/

If you want more background information on this subject, take a look at Eric Smith's **DVD Information** pages, they may be found at
http://www.brouhaha.com/~eric/video/dvd/

Adapting your DVD player for compatibility with a certain region code is not illegal as long it is for private and personal use. Be warned, however, that any modification to the hardware may void your warranty.

(995086-1)

SEL is high. Channel status information is displayed for the channel selected by CS12. C0, which is channel status bit 0, defines professional ($\overline{C0}$= 0)or consumer ($\overline{C0}$= 1) mode and further controls the definition of the Ca-Ce pins. These pins are updated with the rising edge of CBL.

CS12 Channel Select, pin 13
This pin is also dual function and is selected by bringing SEL high. CS12 selects sub-frame 1 (when low) or sub-frame 2 (when high) to be displayed by channel status pins $\overline{C0}$ and Ca through Ce

FCK Frequency Clock, pin 13
Frequency clock input that is enabled by bringing SEL low. FCK is compared to the received clock frequency with the value displayed on F2 through F0. Nominal input value is 6.144 MHz.

E0, E1, E2 Error Condition, pins 4-6
Encoded error information that is enabled by bringing SEL low. The error codes are prioritized and latched so that the error code displayed is the highest level of error since the last clearing of the error pins. Clearing is accomplished by bringing SEL high for more than 8 MCK cycles.

F0, F1, F2 Frequency Reporting Bits, pins 2-3, 27
Encoded sample frequency information that is enabled by bringing SEL low. A proper clock on FCK must be input for at least two thirds of a channel status block for these pins to be valid. They are updated three times per block, starting at the block boundary. These pins are invalid when the PLL is out of lock.

ERF Error Flag, pin 25
Signals that an error has occurred while receiving the audio sample currently being read from the serial port. Three errors cause ERF to go high: a parity or biphase coding violation during the current sample, or an out of lock PLL receiver.

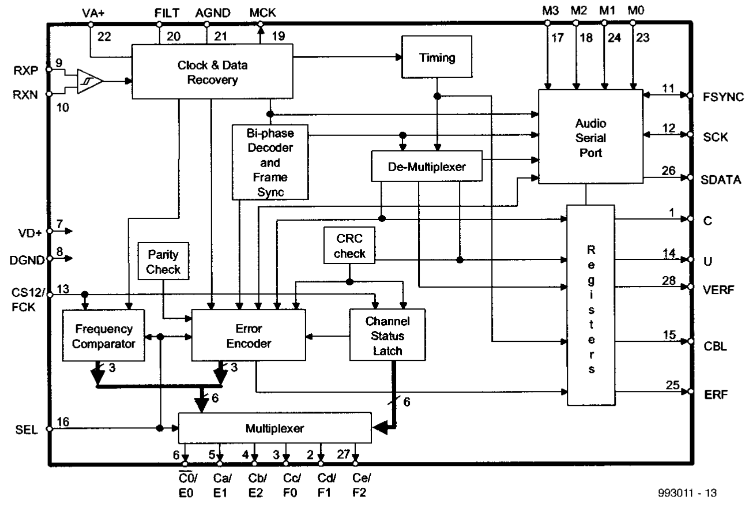
Receiver interface

RXP, RXN Differential Line Receivers, pins 9, 10
RS422 compatible line receivers.

Phase Locked Loop

MCK Master Clock, pin 19
Low jitter clock output of 256 times the received sample frequency.

FILT Filter, pin 20
An external 470Ω resistor and 0.068µF capacitor is required from FILT input to analog ground.



CS8414 Block Diagram.

CS8414
96 kHz Digital Audio Receiver

Manufacturer

CRYSTAL®
A DIVISION OF CIRRUS LOGIC

Cirrus Logic, Inc., Crystal Semiconductor Division,
P.O. Box 17847, Austin, Texas 78760, U.S.A.
Tel. (512) 445 7222, fax (512) 445 7581.
Internet: http://www.crystal.com

Features
- Sample rates to > 100 kHz
- Low-Jitter, On-Chip Clock recovery
- 256xFs Output clock Provided
- Supports: AES/EBU, IEC 958, S/PDIF, & EIAJ
- CP340/1201 Professional and Consumer Formats
- Extensive Error Reporting
- Repeat Last Sample on Error Option
- On-Chip RS422 Line Receiver
- Pin Compatible with CS8411 and CS8412

Ordering information
CS8414-CS 0° to 70° C 28-pin plastic SOIC

Application example
Audio DAC 2000,
Elektor Electronics November 1999.

Description
The CS8414 is a monolithic CMOS device which receives and decodes audio data up to 96 kHz according to the AES/EBU, IEC958, S/PDIF and EIAJ CP340/1201 interface standards. The CS8414 receives data from a transmission line, recovers the clock and synchronization signals, and demultiplexes the audio and digital data. Differential and single-ended inputs can be decoded.
The CS8414 de-multiplexes the channel, user and validity data directly to serial output pins with dedicated output pins for the most important channel status bits. The CS8414 does not need a microprocessor to handle the non-audio data (although a micro may be used with the C and U ports). Instead, dedicated pins are available for the most important channel status bits. The CS8414 is a monolithic CMOS circuit that receives and decodes digital audio data which was encoded according to the digital audio interface standards. It contains an RS422 line receiver and clock and data recovery utilizing an on-chip phase-locked loop. The audio data is output through a configurable serial port that supports 14 formats. The channel status and user data have their own serial pins and the

Normal Audio Port modes (M3 = 0)			
M2	**M1**	**M0**	**Format**
0	0	0	0 – Out, L/R, 16-24 Bits
0	0	1	1 – In, L/R, 16-24 Bits
0	1	0	2 – Out, L/R, I^{2}S Compatible
0	1	1	3 – In, L/R, I^{2}S Compatible
1	0	0	4 – Out, WSYNC, 16-24 bits
1	0	1	5 – Out, L/R, 16 Bits LSBJ
1	1	0	6 – Out, L/R, 18 Bits LSBJ
1	1	1	7 – Out, L/R, MSB Last

Special Audio port Modes (M3= 1)			
M2	**M1**	**M0**	**Format**
0	0	0	8 – Format 0 – No repeat on error
0	0	1	9 – Format 1 – No repeat on error
0	1	0	10 – Format 2 – No repeat on error
0	1	1	11 – Format 0 – Async. SCK input
1	0	0	12 – Received NRZ Data
1	0	1	13 – Received Bi-phase Data
1	1	0	14 - Reserved
1	1	1	15 – CS8414 Reset

validity flag is OR'ed with the ERF flag to provide a single pin, VERF, indicating that the audio data may not be valid. This pin may be used by interpolation filters that provide error correction.

Pin descriptions

Power Supply Connections

VD+ Positive Digital Power, pin 7
Positive supply for the digital section. Nominally + 5 volts.

VA+ Positive Analog Power, pin 22
Positive supply for the analog section. Nominally + 5 volts.

DGND Digital Ground, pin 18
Ground for the digital section. DGND should be connected to the same ground as AGND.

AGND Analog Ground, pin 21
Ground for the analog section. AGND should be connected to the same ground as DGND.

Audio Output Interface

SCK Serial Clock, pin 12
Serial clock for SDATA pin which can be configured (via the M0, M1, M2 and M3 pins) as an input or output, and can sample data on the rising and falling edge. As an output, SCK will generate 32 clocks for every audio sample. As an input, 32 SCK periods per audio sample must be provided in all normal modes.

FSYNC Frame Sync, pin 11
Delineates the serial data and may indicate the particular channel, left or right, and may be an input or output. The format is based on M0, M1, M2 and M3 pins.

SDATA Serial Data, pin 26
Audio data serial input pin

M0, M1, M2, M3 Serial Port Mode Select, pins 23, 24, 18, 17
Selects the format of FSYNC and the sample edge of SCK with respect to SDATA. M3 selects between eight normal modes (M3= 0), and six special modes (M3= 1).

Control Pins

VERF Validity + Error Flag, pin 28
A logical OR'ing of the validity bit from the received data and the error flag. May be used by interpolation filters to interpolate through errors.

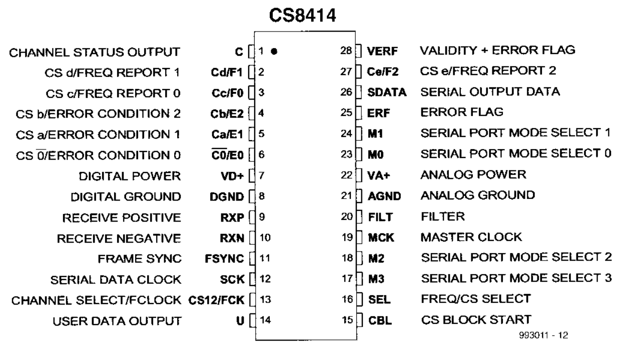
U User bit, pin 14
Received user bit serial output port. FSYNC may be used to latch this bit externally. (Except in I^{2}S modes when this pin is updated on the active edge of FSYNC).

C Channel Status Output, pin 1
Received channel status bit serial output port. FSYNC may be used to latch this bit externally. (Except in I^{2}S modes when this pin is updated on the active edge of FSYNC).

CBL Channel Status Block Start, pin 15
The channel status block output is high for the first four bytes of channel status and low for the last 20 bytes.

SEL Select, pin 16
Control pin that selects either channel status information (SEL = 1) or error and frequency information (SEL = 0) to be displayed on six of the following pins.

$\overline{C0}$, Ca, Cb, Cc, Cd, Ce Channel Status Output Bits, pins 2-6, 27
These pins are dual function with the 'C' bits selected when


CS8414
CHANNEL STATUS OUTPUT C 1
CS d/FREQ REPORT 1 Cd/F1 2
CS c/FREQ REPORT 0 Cc/F0 3
CS b/ERROR CONDITION 2 Cb/E2 4
CS a/ERROR CONDITION 1 Ca/E1 5
CS $\overline{0}$/ERROR CONDITION 0 $\overline{C0}$/E0 6
DIGITAL POWER VD+ 7
DIGITAL GROUND DGND 8
RECEIVE POSITIVE RXP 9
RECEIVE NEGATIVE RXN 10
FRAME SYNC FSYNC 11
SERIAL DATA CLOCK SCK 12
CHANNEL SELECT/FCLOCK CS12/FCK 13
USER DATA OUTPUT U 14
28 VERF VALIDITY + ERROR FLAG
27 Ce/F2 CS e/FREQ REPORT 2
26 SDATA SERIAL OUTPUT DATA
25 ERF ERROR FLAG
24 M1 SERIAL PORT MODE SELECT 1
23 M0 SERIAL PORT MODE SELECT 0
22 VA+ ANALOG POWER
21 AGND ANALOG GROUND
20 FILT FILTER
19 MCK MASTER CLOCK
18 M2 SERIAL PORT MODE SELECT 2
17 M3 SERIAL PORT MODE SELECT 3
16 SEL FREQ/CS SELECT
15 CBL CS BLOCK START
993011 - 12

CS8414 Pinout.

Absolute maximum ratings (GND = 0V, all voltages with respect to ground)

Parameters	Symbol	Min.	Max.	Unit
Power Supply Voltage	VD+ , VA+	-	6.0	V
Input Current, Any Pin Except Supply (Note 1)	I_{in}	-	±10	mA
Input Voltage, Any Pin Except RXP, RXN	V_{IN}	–0.3	(VD+) + 0.3	mA
Input Voltage, RXP and RXN	V_{IN}	–12	12	V
Storage Temperature	T_{stg}	–65	150	°C

Notes: 1. Transient currents of up to 100 mA will not cause SCR latch-up

Recommended operating conditions (GND = 0V, all voltages with respect to ground)

Parameters	Symbol	Min.	Typ.	Max.	Unit
Power Supply Voltage	VD+ , VA+	4.75	5.0	5.25	V
Supply Current VA+	I_A	-	20	30	mA
Supply Current VD+	I_D	-	20	30	mA
Ambient Operating Temperature: (note 2)	T_A	0	25	70	°C
Power Consumption	P_D	-	175	315	mW

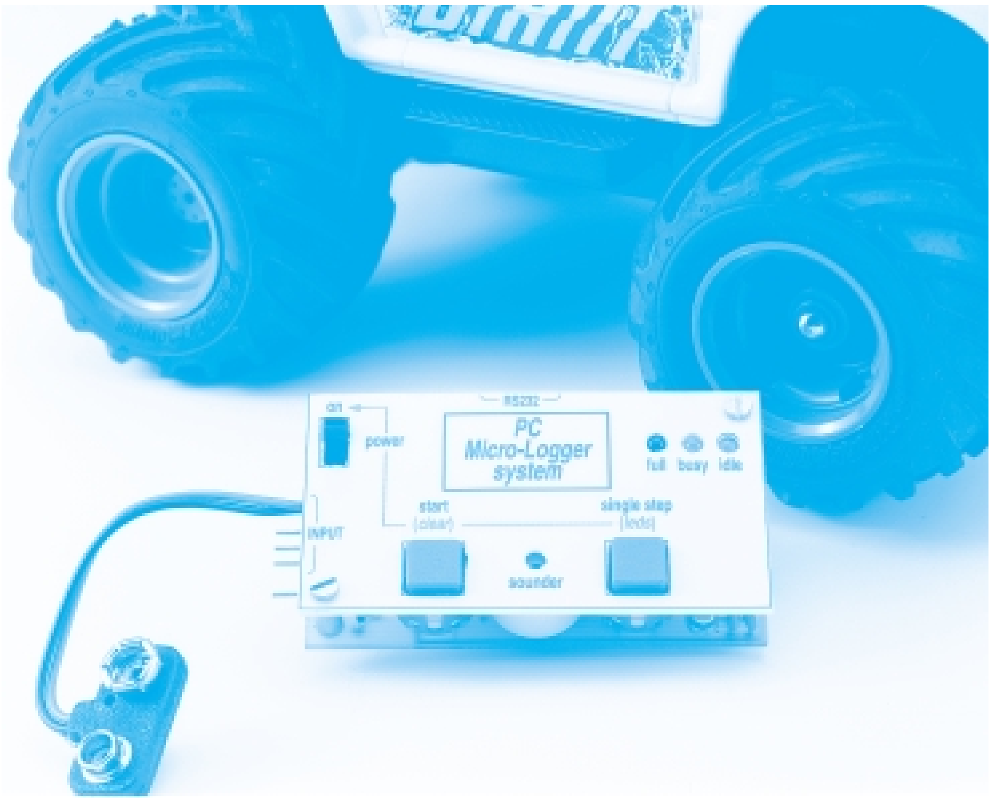
Notes: 2. The -CS parts are specified to operate over 0 to 70°C but are tested at 25°C only.

Digital characteristics (TA = 25°C; VD+ , VA+ = 5V ±5%)

Parameters	Symbol	Min.	Typ.	Max.	Unit
High-Level Input Voltage, except RXP, RXN	V_{IH}	2.0	-	-	V
Low-Level Input Voltage, except RXP, RXN	V_{IL}	-	-	+ 0.4	V
High-Level Output Voltage (I_O = 200μA)	V_{OH}	(VD+) – 1.0	-	-	V
Low-level Output Voltage (I_O = –3.2mA)	V_{OL}	-	-	0.5	V
Input Leakage Current	I_{in}	0	1.0	10	μA
Input Sample frequency (Note 3)	F_S	28.4	-	100	kHz
Master Clock Frequency (Note 3)	MCK	7.28	256xF_S	25.6	MHz
MCK Clock Jitter	t_j	-	200		psRMS
MCK Duty Cycle (high time/cycle time)		-	50	-	%

Notes: 3. F_S is defined as the incoming audio sample frequency per channel.

The micrologger presented here is remarkable because this device, no bigger than a matchbox, is a data logger and voltmeter that can operate fully on its own. In addition, it has an RS232 port for connection to the COM port of a PC. Commands to be executed by the micrologger's built-in command processor can be entered via the keyboard of the PC, and the measurement result is displayed on the screen of the monitor. After being configured via the PC, the micrologger can be disconnected and used stand-alone. It can store up to 82 10-bit measurement values in EEPROM. Stored measurements are saved even when the power is switched off. After the micrologger has again been connected to the PC, the measurement data can be uploaded and converted into a graphic display, using a spreadsheet program such as KyPlot.



Design by B. Stuurman

micro datalogger

for use with a PC and stand-alone

Micrologger specifications

- operates as a standalone data logger and 'single-stepper' with data postprocessing on a PC, as well as a PC voltmeter/datalogger and 'single-stepper'
- non-volatile memory for 82 3-digit samples
- sampling delay adjustable from 1 second to 15:59:59 h:m:s
- maximum measurement session duration approximately 1300 hours
- RS232 interface; protocol 38400 (19200), 7, n, 2
- screen update rate 3x per second
- range / basic accuracy

range	basic accuracy
0 to 2.49 V	±(0.5%, 1 d)
2.5 to 9.99 V	±(0.5%, 3 d)
10.0 to 99.9 V	±(0.5%, 2 d)

- input impedance: approx. 500 kΩ
- power supply: 9-V battery
- current consumption: 4.6 mA (LEDs on); 2.8 mA (LEDs off)
- dimensions (exclusive of protruding connector pins): 64×36×15 mm (l × w × h)
- weight without battery: approx. 30 g

Its small dimensions and stand-alone operation capability make the micrologger an outstanding choice for 'remote' measurements. For example, it could be used to measure the voltage and current consumption of the motor of an electrically driven model, to allow the drive components to be optimized. Amateur meteorologists could collect measurements over a long period of time without having to be personally present. Two data loggers could be carried aloft by a weather balloon, one to measure the temperature and the other to measure the altitude. In physics exercises in which several teams take measurements, each team could be provided with a micrologger to store their data. The advantage of this approach is that

an expensive collection of instruments is not needed to perform the measurements, and at the same time all measurements are registered by team and can be subsequently processed.
When the micrologger is connected to a PC, the command 'volt' causes the value of the actual voltage to be sent to the PC three times per second, via the RS232 interface. The PC screen acts as the display, and if a 'full-screen' character size is used, the display can be read at a distance of several metres.

An introduction

Figure 1 shows the complete schematic of the micrologger. At its heart is an ST62T60, a powerful 8-bit microcontroller from SGS-Thomson. The following peripheral components are connected to the core of the controller via an internal serial data bus:

- a normal timer,
- a timer with a pulse-width modulated (PWM) output,
- a watchdog timer,
- an 8-bit A/D converter,
- a serial interface (Serial Peripheral Interface, or SPI),
- 128 bytes of nonvolatile memory (EEPROM).

The normal timer is always active and looks after the timing of several different software processes. This takes place in the interrupt routine by means of timer flags. Since this timer is also used to generate the seconds flag, it is always loaded with a fixed value that yields a 'nearly perfect' seconds clock after some further division.
The PWM timer is used to generate a pulse waveform signal (on the ARTIM output). The average value of this signal is used to increase the resolution of the 8-bit A/D converter to 10 bits. We'll say more about this shortly.
The watchdog timer is used for serial I/O timing, to free up the other timers. This is possible since the watchdog timer can be activated under software control. Except for this timer and the SPI, no hardware is used for the basic I/O function.
The 8-bit A/D converter is naturally used for voltage measurements. To be precise, the voltage on the analogue input (Ain) is measured 64 times for each sample, and the result is then divided by 64. The range of the A/D converter is from 00_H to FF_H. The first of these values will be measured if the voltage on Ain is equal to the voltage on the Vss pin (0V), and the second value will be measured if the voltage on Ain is equal to voltage on the Vdd pin (+5V). The accuracy of the micrologger is thus directly related to the accuracy of the supply voltage (Vdd). For this reason, an LP2950CZ5.0 has been chosen for the voltage regulator. Not only is this a low-drop regulator, it is also recommended as a reference source by its manufacturer!
Current sourcing or sinking by other components in the circuit can affect the conversion, due to voltage drops in the wiring. For this reason, the LEDs are switched off while a measurement is being made. This produces a characteristic blinking effect.
The serial interface serves for communication with a PC. It allows commands to be sent to the micrologger, and data to be sent from the micrologger to the PC. The micrologger contains a true interactive command processor. Of course, a terminal emulator program, such as HyperTerminal (supplied with Windows), must be present on the PC.
The 128-byte EEPROM is used to store various settings, including the sampling delay parameter, and it also includes room for 82 11-bit sample values (10 data bits and a decimal point). The samples are written in an encoded form, with two samples stored in every three bytes of memory to make the best use of the available capacity.

More details

The RS232 interface

The ability to communicate with a PC via the RS232 interface is vitally important for this application. However, this creates

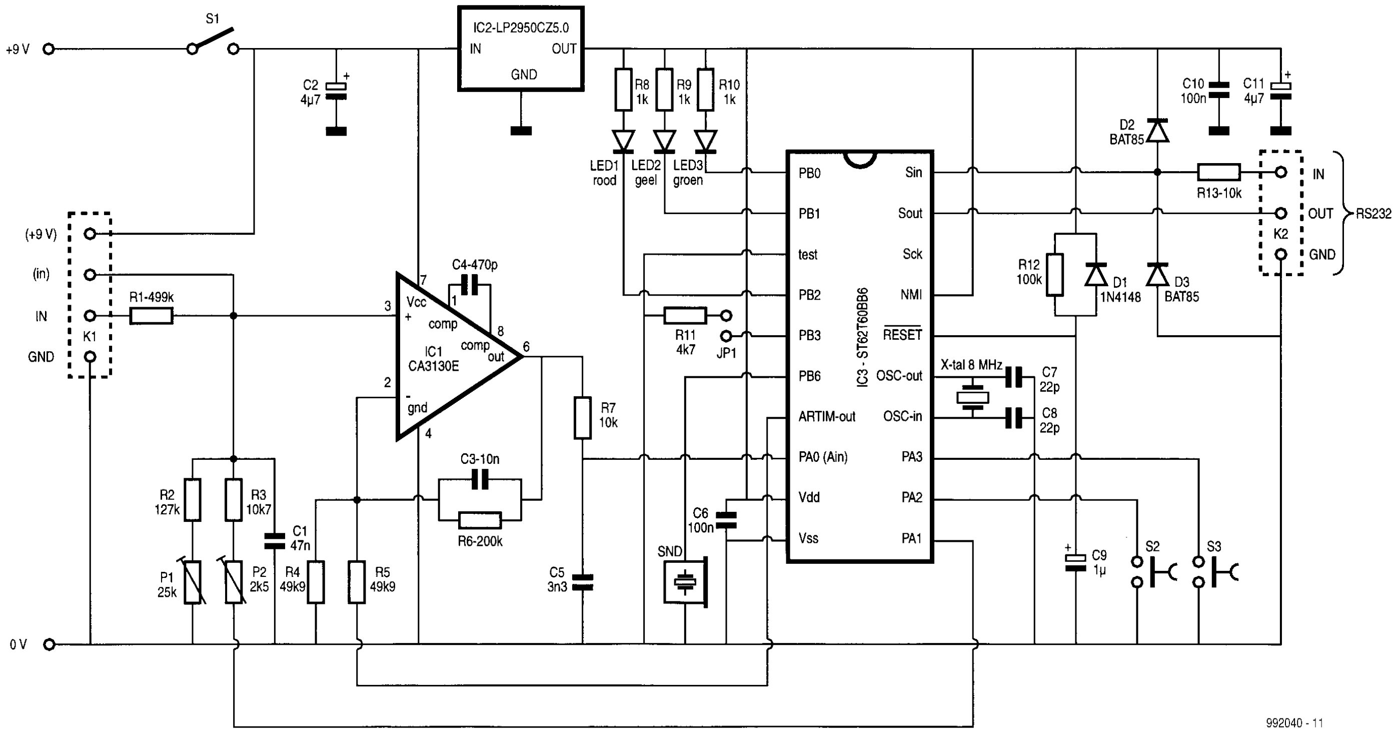


Figure 1. A microcontroller, an opamp and a handful of components are all you need to build a mini PC data logger and voltmeter.

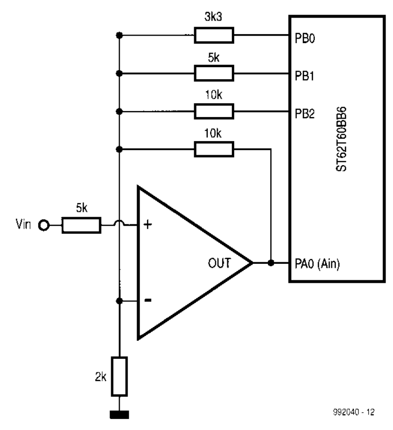


Figure 2. Using an opamp as an adder amplifier to increase the resolution from 8 to 10 bits.

a problem, since this communication must not be allowed to disturb the normal operation of the micrologger.

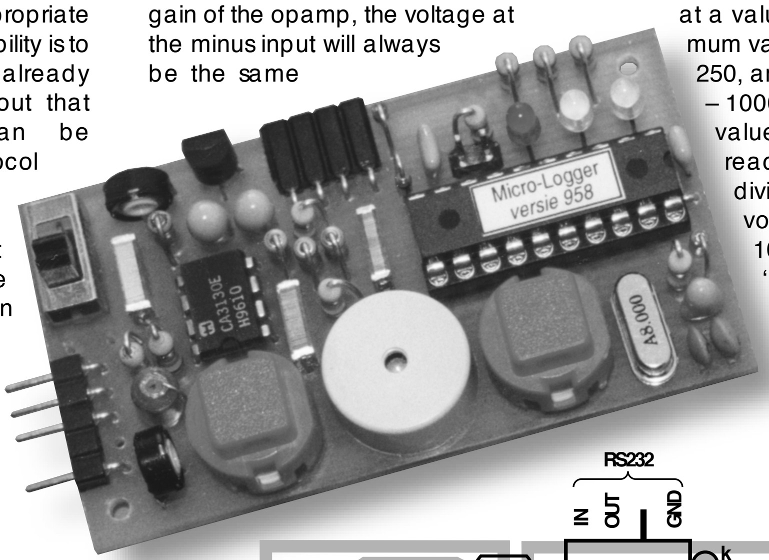
Usually, an RS232 interface is built up using software, in which the baud rate is determined by delay loops, or with the aid of timers. However, here the timers are indispensable for other purposes, and the software approach is only possible for relatively low baud rates, which would not be appropriate here. The only remaining possibility is to use the SPI, which after all is already present. Ultimately, it turns out that RS232 communication can be achieved by using a 7-bit protocol and look-up tables. During normal operation, the SPI is set up to receive data. A start bit at SIn (see **Figure 1**) triggers the SPI, which subsequently takes in the data all on its own. On completion, the SPI generates an interrupt, which is used to set a flag. This flag is regularly polled in the main program loop, which can take the necessary actions if it is set. The hardware for the RS232 interface is minimal — two Schottky diodes (for speed) and one resistor. The only thing that is necessary is that the connection cable includes a number of cross-connections that serve to 'trick' the UART of the PC (see the January 1999 issue of Elektor for more information). The maximum speed of the SPI is very high, but due to the clock rate of 8MHz and the available predivider settings, only 38400 baud and 19200 baud are actually possible. The latter value can be selected by installing jumper JP1.

From 8-bit to 10-bit ADC resolution

Although the resolution of the A/D converter of the ST62T60 is 8 bits, it is possible to increase this resolution with an opamp. The basic idea, illustrated in **Figure 2**, comes from an ST application handbook [1].

The opamp is wired as a noninverting amplifier with negative feedback. Due to the negative feedback and the high gain of the opamp, the voltage at the minus input will always be the same as the voltage at the plus input. When Vin reaches a level that causes the A/D converter to produce its maximum output value, PB2 is set high. The resistor values are chosen so that the output voltage of the opamp then goes to zero, which allows the voltage on Vin to continue to increase until the A/D converter again reaches its maximum output value. When this happens, PB1 is set high and the opamp's output voltage again goes to zero. The measurement range of the A/D converter is thus used four times over, so that the resolution is increased from 8 to 10 bits.

Now let's return to **Figure 1**. The CA3130 opamp has two special properties, which are that its inputs can handle voltages down to just below earth level and that the output voltage range extends down to the earth level. This opamp is connected directly to the +9V supply, so that its output has no difficulty reaching 5V, which is the maximum input level of the A/D converter. The PWM signal (ARTIM out) is connected to the minus input of the opamp via R5. There are four working ranges: 0%, 25%, 50% and 100%. To simplify the calculations, the switching points have been chosen to occur at a value of 251 (FB_H). The maximum value for each range is thus 250, and the total range is thus 0 – 1000. The software tests for a value of 1000 and, if this is reached, switches in a 10x divider by setting PA1 low. For voltage readings under 1000, P1 is configured as a 'floating' input. The voltmeter thus has two



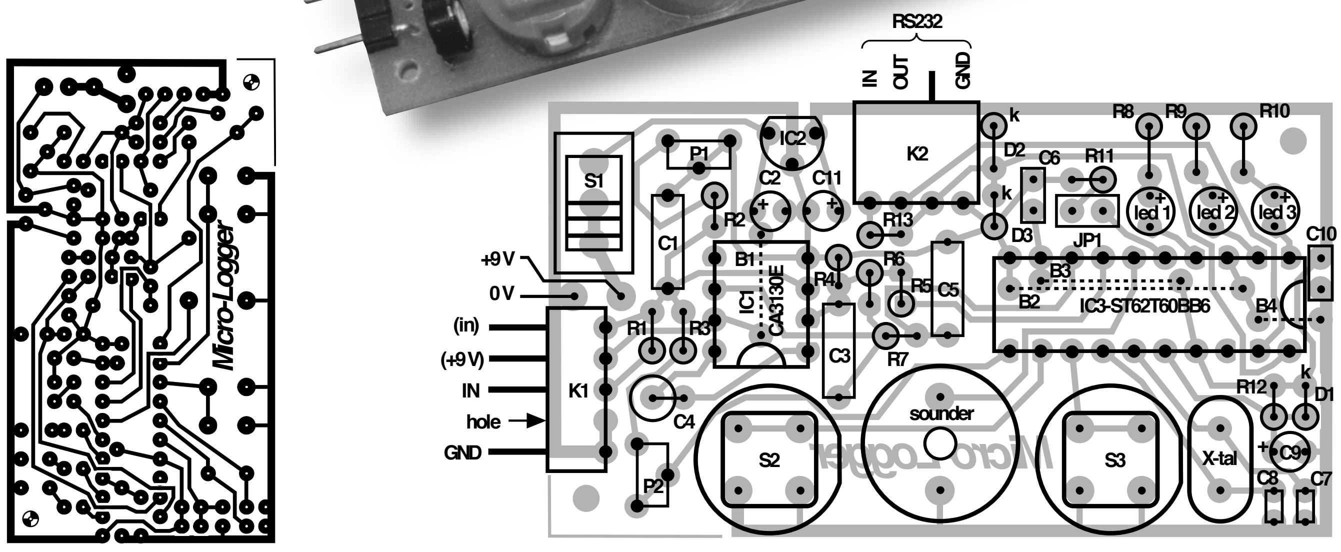
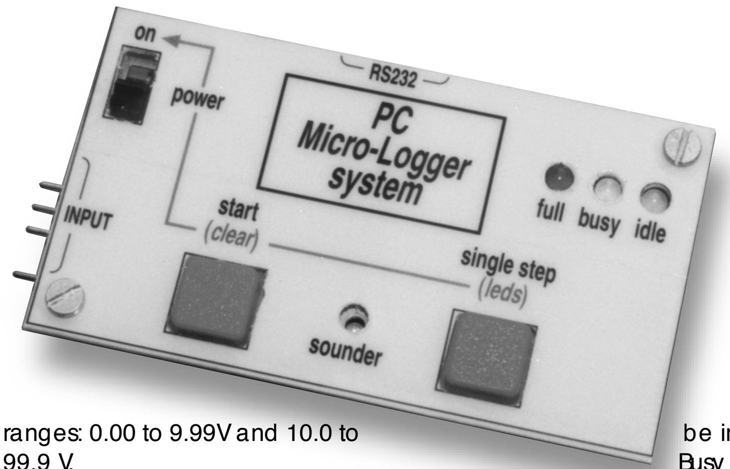


Figure 3. The circuit board layout (scale 1:1) and component layout of the micrologger (scale 2:1).



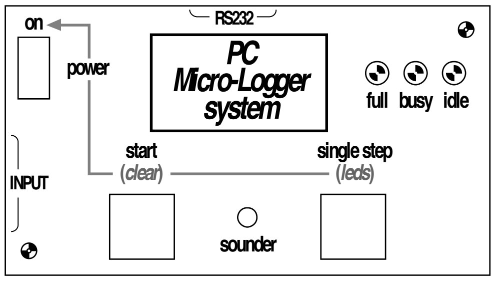


Figure 4. Sample front panel for the micrologger.

ranges: 0.00 to 9.99 V and 10.0 to 99.9 V.

A pulse waveform at around 31 kHz from the ARTIM output is injected into the minus input of the opamp. Filter capacitors C3 and C5 are provided to extract the average value of this signal. The input of the micrologger is K1. In addition to the usual IN and GND inputs, a direct input (in) and the 9 V supply voltage are present at K1. The opamp circuit has a gain of around 9, and a voltage of around 2.2 V on the direct input results in an output voltage of 9.99 V. This means that an interface circuit can be connected to K1, for example for making current or temperature measurements.

The LEDs and S1, S2 and S3

The LEDs indicate the status of the micrologger. If no samples are being taken and the data memory is empty, the green LED will be on (Idle). If the memory contains sample data but is not full, the yellow LED will be on (Busy). When the memory is full, the red LED will be on (Full). The state of the LEDs is stored in the EEPROM. If the micrologger is switched on when the memory already contains sample data, this will be indicated by the Busy or Full LED being on. In this case, it is not possible to start sampling or to single step the micrologger. The memory must first be cleared. S1 is the on/off switch. If S2 is pressed while the Idle LED is on, the micrologger starts collecting data and no longer responds to S2 or S3. If S3 is pressed while the Idle LED is on, the micrologger takes a single sample of the actual voltage, and it no longer responds to S2. S3 may be pressed up to 82 times in succession.

If S2 is held pressed while the micrologger is switched on with S1, the memory is cleared. If S3 is held pressed while the micrologger is switched on with S1, the LEDs are switched on or off. Switching the LEDs off substantially reduces the current consumption of the micrologger. Although this setting is always saved in the EEPROM, there is one exception. If the LEDs have been switched off and samples are collected, the Full LED will be switched on when the memory is full.

Signals

The micrologger beeps after each sample has been taken, and after 82 samples — that is, when the memory is full — it beeps six times. If the memory is partially or completely filled and the micrologger is switched off and then switched on again, for example in order to read out the sample data to a PC, it will beep three times if you try to start sampling or to operate it in single-step mode (via the PC or using its push-buttons). The same thing happens if a 'step' command is issued via the PC while the micrologger is sampling, or vice versa. A new sampling or single-step session can be started only after the memory has been cleared.

If the micrologger is operated interactively with a PC, it beeps once if it receives an invalid input. This can be an incorrect command or an invalid parameter.

Building the micrologger

Figure 3 shows the printed circuit board and component layouts. Start with the four wire jumpers, and then mount the rest of the components. K1 is a male header with one more pin than is needed. To prevent incorrect mating of K1 with its matching connector, remove the 'extra' pin at the indicated

COMPONENTS LIST

Resistors:
(All resistors 0.25 or 0.2 W)
R1 = 499kΩ 1%
R2 = 127kΩ 1%
R3 = 10.7kΩ 1%
R4,R5 = 49.9kΩ 1%
R6 = 200kΩ 1%
R7,R13 = 10kΩ
R8,R9,R10 = 1kΩ
R11 = 4.7kΩ
R12 = 100kΩ
P1 = 25kΩ preset, 6 mm, vertical
P2 = 2.5kΩ preset, 6 mm, vertical

Capacitors:
C1 = 47nF, MKT, raster 7.5mm
C2,C11 = 4.7μF 15V, tantalum
C3 = 10nF, MKT, raster 7.5mm
C4 = 470pF, polystyrene (Siemens)
C5 = 3.3nF, MKT, raster 7.5mm
C6,C10 = 100nF, raster 0.1", multilayer
C7,C8 = 22pF, raster 0.1", ceramic
C9 = 1μF 15V, tantalum

Semiconductors:
IC1 = CA3130E
IC2 = LP2950CZ5.0
IC3 = ST62T60BB6 (programmed, order code 996518-1)
D1 = 1N4148
D2,D3 = BAT85
Led1 = LED 3 mm, high intensity, red
Led2 = LED 3 mm, high intensity, yellow
Led3 = LED 3 mm, high intensity, green

Miscellaneous:
S1 = 1-pole slide switch, 11 mm, PCB mount (Knitter type MFP 120)
S2,S3 = pushbutton, square (D6)
X-tal = quartz crystal 8 MHz, HC-49/S (height approx. 4 mm)
Snd = piëzo-ceramic buzzer, 14 mm diam., raster 7 mm
K1 = angled pinheader, SIL, 5 contacts
K1-contra = straight socket, SIL, 5 contacts
K2 = angled socket, , SIL, 4 contacts
K2-contra = straight pinheader, SIL, 4 contacts
JP1 = straight pinheader, 2 contacts
JP1-contra = jumper (shunt), 2.54 mm
B1-B4 = wire link
20-pin IC socket
Battery clip for 9-V PP3 battery
3.5"-floppy disk, contains project source-code: order code **996026-1**

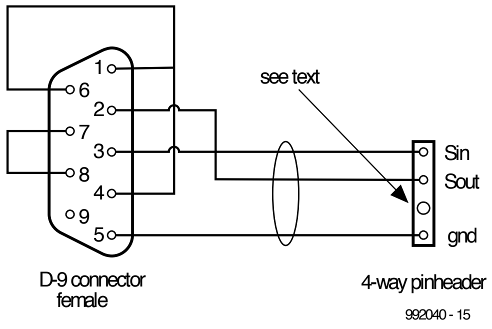


Figure 5. Cable for connecting the micrologger to the RS232 port of a PC. The D-type connector includes several cross connections to 'trick' the UART.

location and enlarge the hole using a 1.4-mm drill. Then glue a pin at the same location on the matching connector, using a small drop of crazy glue. Now the connectors can never be mated the wrong way around.
Treat K2 and its matching connector in the same manner. Finish off the leads of the matching connectors with heat-shrink tubing to give a neat appearance. Make sure that the flats of S2 and S3 face to the right.
Finally, you can copy the front panel layout in **Figure 4**, cut it out and glue it to a 1-mm thick piece of ABS plastic. After you have made the necessary openings, you can attach this panel to the circuit board using standoffs and 2-mm screws.

Software

Before the micrologger can be connected to the COM port of a PC, a cable must be constructed as shown in **Figure 5**. The 9-way plug can be connected either directly to the COM port or via a serial extension cable, with a 25-way connector at the PC end if necessary.
The HyperTerminal program can be used for communication with the PC. Start the program and select File/Properties. Set Connect to 'Direct to COMn', where n is number of the COM port used. Now choose Configure and enter: 38400, 7, none, 2, flow control = 'Hardware'. Select the Settings tab card and set Emulation = 'Auto detect'. Select ASCII Setup and make sure that only 'Wrap lines' is checked.
Save this file on the desktop as Logger.ht, or with any other suitable name (as long as it has the '.ht' extension). Close HyperTerminal and then start it up again, using the new icon on the desktop. Only then will the new settings take effect!
If the micrologger is not yet connected to the PC, connect it now and switch it on.
Tip: use only lower-case characters for communicating with the micrologger!
Press Enter. The micrologger will beep and respond with 'command?'. Now press '?' and then Enter. The micrologger will respond with a list of all available commands.
The delay time can be set by entering it following the 'delay' command. To see what the current delay time setting is (and its format), simply type 'delay'.
The commands 'delim(iter)' and 'd(ecimal)p(oint)' change the formatting of the export file.
The remaining commands are self-explanatory, although there's one handy tip. If the 'volt' command is active, it can be switched off by pressing the Escape key. This can also be used to nullify any unintentionally entered command.
The sample data in the memory are exported in response to '^ d' (Control d). Before doing this, you can prepare a file to receive the data by selecting Transfer and then 'Capture text' in HyperTerminal, and giving the file a name with the extension '.txt'. The exported data will then be directly written to the specified file. Be sure to close this file before issuing any new commands. Alternatively, the samples can be selected on the screen and transferred via the clipboard to the notepad, for example.
The file data can be read directly into the KyPlot spreadsheet program [2], so that the data can be presented in graphic format.

Calibration

Individual examples of the microcontroller show slight differences in behaviour which, small as they may be, make it impossible to specify fixed values for the breakpoints at 25%, 50% and 75%. A simple but effective solution to this problem is found by including commands that allow these values to be adjusted; these are the 'inc' and 'dec' commands. You will need a good DVM and a variable power supply (0 to 15 V) to perform the calibration. If you have access to an oscilloscope, it's a good idea to connect it to the ARTIM output (pin 7). Using the oscilloscope, you can readily see where the breakpoints lie, and it is thus easier to adjust them for smooth transitions.
Connect both the DVM and the micrologger to the variable power supply, and adjust the voltage to approximately 2.00 V. Start HyperTerminal and enter the command 'volt'. Now adjust P1 until the reading on the screen is the same as that displayed by the DVM (tip: select a fairly large character size in

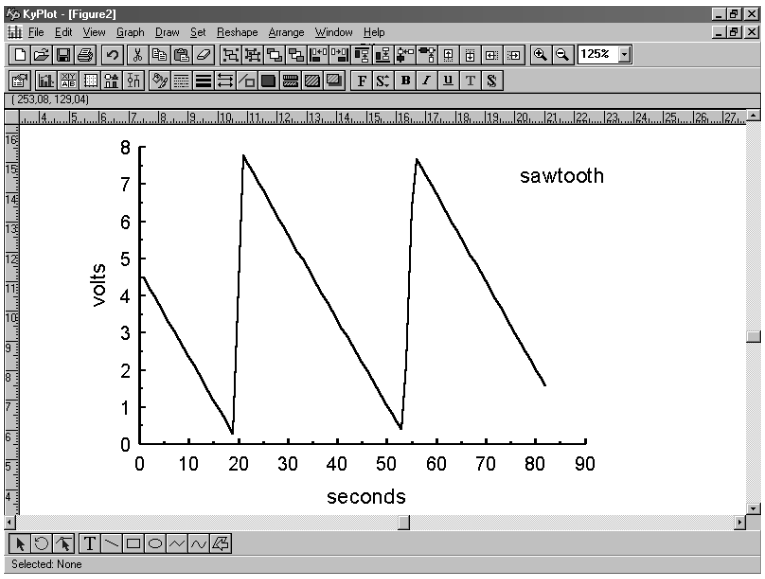


Figure 6. The file containing the measurement data can be loaded into KyPlot in order to directly create a graphic display of the data. Here you see a slow sawtooth waveform produced by a function generator.

HyperTerminal, so it's easy to read).
Now set the voltage to 4.00 V and use the commands 'dec 0' and 'inc 0' to adjust the value displayed on the screen until it is as close as possible to the DVM value. Decrease the voltage to around 2.4V, increase it very slowly to 2.6V and then reduce it again to 2.4V. The crossover at the breakpoint should be as smooth as possible.
Repeat this procedure with a voltage of 6.00 V and the commands 'dec 1' and 'inc 1' (breakpoint 5.0 V), and then with 8.00 V and the commands 'dec 2' and 'inc 2' (breakpoint 7.5 V).
Now adjust the voltage so the value displayed on the screen jumps between 9.99 V and some other value. Adjust P2 so that the 'other value' is 10.0 V (and not 09.9 V).
This completes the calibration process.

Conclusion

The accuracy of the clock should be more than adequate in most cases. It can be checked by setting the delay time to one hour. Wait until the seconds hand of your watch reaches 12, and start the micrologger. It will emit a beep then, and every hour thereafter, up to 82 times. Each time a beep occurs, the seconds hand of your watch will indicate the extent to which the micrologger clock runs fast or slow. If necessary, the clock rate can be adjusted by experimenting with the values of C7 and C8.

The SPI is used for both reception and transmission. When the 'volt' command is active, it can happen that SPI is transmitting at exactly the same time as a key is pressed, in which case the key code will be lost. If this happens, press Enter and then enter the command once again.

(990062)

Literature

[1] ST Application Note AN420/0294, 'Expanding A/D Resolution of the ST6 A/D converter'.
[2] KyPlot http://www.qualest.co.jp/Download/KyPlot/kyplot_e.htm (see Elektor Electronics March 1999 issue).

Anyone who carries around every type of connection cable together with a laptop computer, often finds that the majority of the baggage is cables instead of the laptop itself. However, this size of this cable salad can easily be minimized. The standard cables described here, together with a few gender changers, support all possible connections.

By H.-J. Böhling

a cable for all occasions

correct connections for your laptop PC

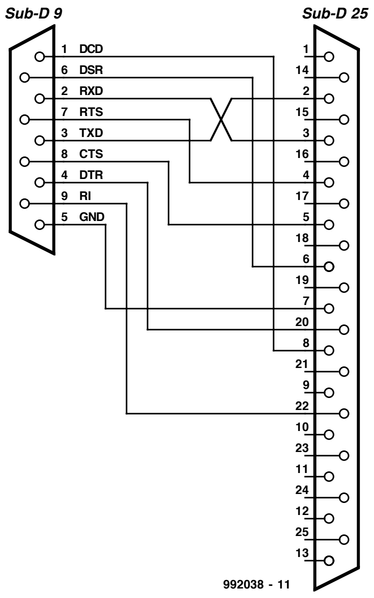


Figure 1. 9-way to 25-way RS232 adapter.

You really need a sort of Swiss army knife of the cable world to be prepared for all imaginable serial and parallel connections. The set of cables and adapters described in this article will handle all possible combinations of connections. The basic cable is an extension cable for the printer port. If you buy this ready-made, make sure that it includes the full set of 25 leads, connected one to one. This cable should be as long as necessary, but no longer than it has to be.
In addition, you need a set of adapters that convert from 9-pin to 25-pin connectors (and the other way around, of course), and others that convert a plug into a socket or a socket into a plug — the so-called *gender changers*. These are commercially available in the form of adapter connectors, which to be sure are practical and light, but they make for quite awkward and unreliable

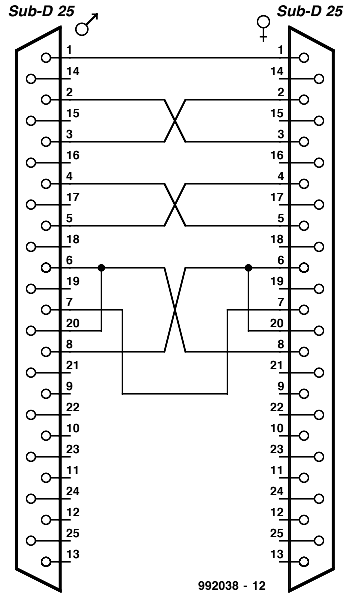


Figure 2. Null modem adapter.

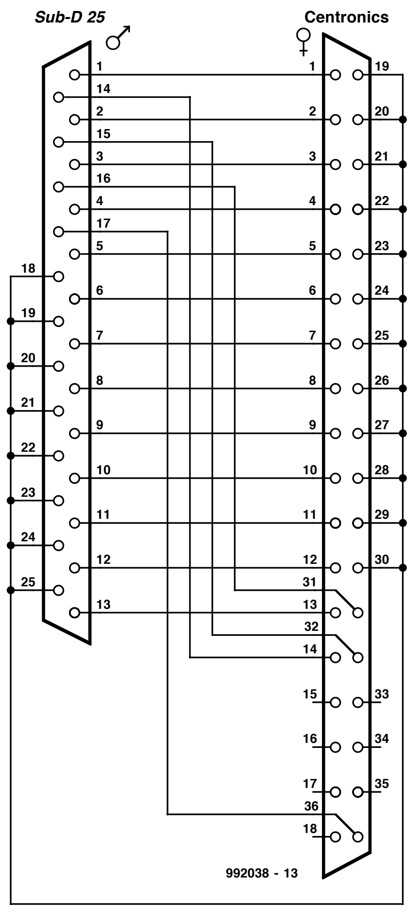


Figure 3. Centronics adapter.

connections when several adapters have to be plugged together in series. It is thus better to use short cables (around 20 cm long) for these connections.

The following is recommended as a basic set:

- ✗ cable, 25-way Sub-D plug to 25-way Sub-D socket, approx. 2 m, 25 leads 1:1;
- ✗ cable, 9-way Sub-D plug to 25-way Sub-D socket, approx. 20 cm, (**Figure 1**);
- ✗ cable, 9-way Sub-D socket to 25-way Sub-D plug, approx. 20 cm, (**Figure 1**);
- ✗ mini gender changer, 25-way Sub-D socket/socket
- ✗ mini gender changer, 25-way Sub-D plug/plug;
- ✗ mini gender changer, 9-way Sub-D socket/socket;
- ✗ mini gender changer, 9-way Sub-D plug/plug;
- ✗ mini gender changer, 25-way Sub-D socket/plug, null modem (**Figure 2**);
- ✗ gender changer, 25-way Sub-D plug/Centronics socket (**Figure 3**).

If you really want to be prepared for everything, you can also add a 25-way sub-D jumper box to this list.

The best-known interconnection method for transferring data between a laptop and a desktop computer is probably DOS 6.x Interlink. Although Interlink can be used with a serial connection, the maximum data transfer rate is only around 35 MB per hour. The data transfer rate via a parallel connection is significantly higher. If you want to be able to enjoy high-speed data transfers, you should without fail prepare an Interlink adapter as shown in **Figure 4**, and fit it into an empty Sub-D adapter housing. This adapter allows connections to be made using not only Interlink under DOS, but also Laplink, Norton Commander (V4.0 and V5.0) and Windows 95 Direct Cable Connection. The data transfer rate ranges between 144 MB and 252 MB per hour, depending on the actual ports used. If you want to learn more on this subject, have a look at the article 'A Simple PC Network' in the PC Topics Supplement of the February 1998 issue of *Elektor Electronics*. Also, the LPT/COM tester to be described in the December 1999 issue forms an ideal supplement to the cable set, since it allows you to 'snoop' within the existing connections.

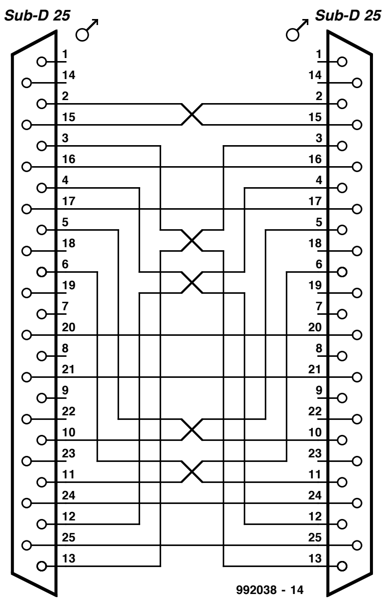


Figure 4. Interlink adapter.

(992038-1)

Having looked last month at the ways in which we can design FPGAs, and coming down on the side of VHDL as a design method, we need to see how we can take our VHDL and turn it into a format we can user to program a physical device. This involves a number of steps from entering the design VHDL, turning it into a format our place and route software can understand, and then turn that information into a bitstream which can be used to program the device.

By Gordon Pocock

Atmel FPGA design course (2)

What is on the CD-ROM?

The CD-ROM, presented free with this month's issue of this magazine, has a number of utilities and tools for the design of FPGAs. Here is a list of the actual contents of this CD-ROM:

HDL Planner is a template generator and text editor for VHDL entry methods. HDL Planner also integrates Atmel's Macro Generator software and is especially useful to the newcomer to VHDL.
Macro Generator software which allows the building of functional blocks which are fully specified and can be reused many times in one or several designs.
Place And Route routines which take the design files and actually perform the layout of the design as implemented on the silicon. The Place and Route routines can also be timing driven to improve design performance.
Static timing analyser used to check path delays through a design.
Bitstream Generation and **Download routines** compatible with all configuration modes.
Everest VHDL Synthesis tool
All the items above are installed when you install the IDS6.0 CD. You can also select additional libraries for other third party tools as listed below.
Libraries for the following Third Party Design Capture Systems:
- Exemplar
- FpgaExpress
- MTI
- OrCad
- Synplicity
- Viewlogic WorkView Office

Doulos Tutorials covering aspects of the VHDL language. Doulos are a training company running public and company based courses on FPGA Design Methodologies and Techniques, as well as offering computer-based courses on VHDL and Verilog.

It may be useful to follow the design flow by installing the IDS6.00 software which will be used to demonstrate the design flow. Although there will not be room within this article to go through in detail every step of the process, this is all covered in the tutorial and documentation files on the CD-ROM.
To install IDS6.00, simply insert the CD-ROM into your drive, and it should automatically run the setup program which will install the software. If the CD-ROM does not autoplay, select Run from the Start menu, and enter:

```
D:\setup.exe
```

Where D: is the drive letter of your CD ROM drive. Follow the on-screen prompts to install IDS 6.00. When asked which libraries to install, you should deselect the Viewlogic option. The Free System that you have been supplied with uses the Everest Design System and the libraries for this are installed by default. To run the Viewlogic system you would need to purchase a license. If there are any installation problems, the process is covered in the documentation files on the CD-ROM. If you experience further problems, contact Kanda Systems, details at the end of the article.
Assuming you have successfully installed IDS 6.00, run the program by selecting *Atmel IDS 6.00* from the Start→Programs menu. After a few seconds you will see a screen as in **Figure 5**. This is the main design environment, and all the other routines, such

as the VHDL template generator, HDL Planner and the VHDL tool EDS are launched from within this environment. The tool bar below the menu bar runs through the design process left to right. We will look at the icons relevant to the VHDL designs we will initially be involved with. A complete description of all the buttons can be found in the on-line documentation and in the book that accompanies this set of articles (available from Kanda Systems Ltd., see advert elsewhere in this issue).

This icon invokes the *Design Launcher*. This gives access to a dialogue that allows the user to set up the environment for a particular design, by specifying, for example, the design name and directory, design capture package and device type. To make use of the accompanying development board, we must always specify the AT40K family of FPGAs, although IDS 6.00 allows the user to design with the AT6000 family of FPGAs also.

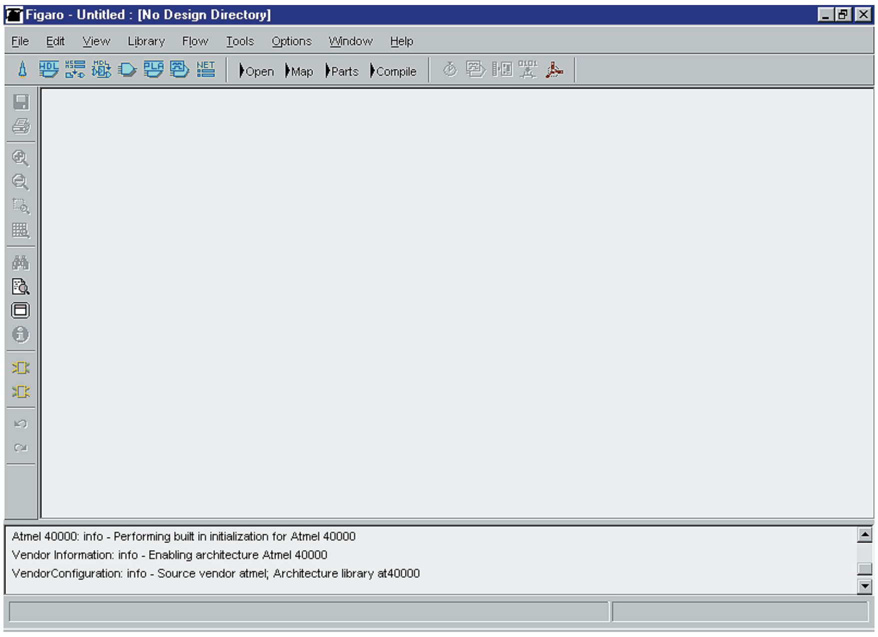
This icon invokes *HDL Planner* (**Figure 6**). *HDL Planner* looks like a simple text editor, but the interface belies its power. *HDL Planner* is an ideal VHDL learning tool in that it will automatically generate the template for a design. This template contains all of the basic VHDL constructs that are required for a design. By using the VHDL pull-down menu, additional, more complex construct templates may be added to the design, again aiding the user by inserting the complete VHDL syntax for a particular type of construct.

HDL Planner is also the interface to the *Macro Generator* functions. These macro generators allow the user to construct fully placed and routed, parameterised components into their design at the VHDL coding stage. This takes away a lot of the task of writing VHDL for standard functions ranging from memories, through counters to multipliers. All of the available macros may have their parameters set by the user. For example, a RAM block may be defined as having a width of *x* and a depth of *y*, whilst a counter would have its reset parameters and count width specified.

This icon allows direct access to the *Macro Generators*. This allows the generation of macros externally from HDL Planner, but is probably most useful as a browser to see what functions are available as macros, and which parameters may be specified. Note that not all of the possible functions in the Macro Generators dialogue are available via HDL Planner. A typical Macro Generator screen is shown in **Figure 7**.

This icon brings us on to the conversion from VHDL to AT40K Library elements. *Everest Design System (EDS)* takes in VHDL code in ASCII text, and converts this into something known as EDIF. EDIF is a generic format defined to allow the transfer of electronic designs between design packages. The process of translation from an ASCII text file to (in this case) an EDIF output is known as Synthesis. This is because EDS takes in a description of a function at a high level, and synthesises the logic required to implement this function in terms of a specific architecture.

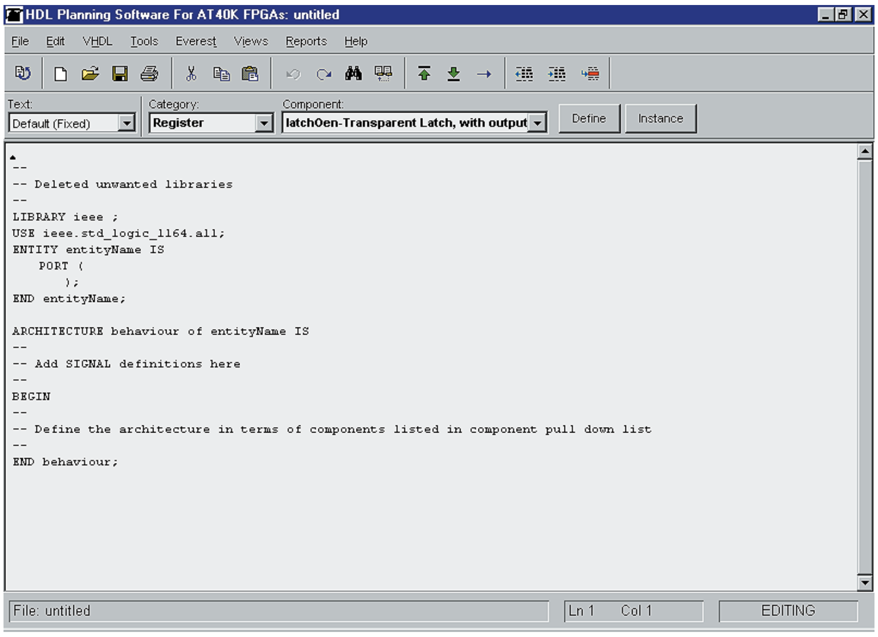
There are several stages to the synthesis of a design by EDS, as shown in **Fig-**



Figaro - Untitled : [No Design Directory]
File Edit View Library Flow Tools Options Window Help
Open Map Parts Compile

```
Atmel 40000: info - Performing built in initialization for Atmel 40000
Vendor Information: info - Enabling architecture Atmel 40000
VendorConfiguration: info - Source vendor atmel; Architecture library at40000
```

Figure 5. Main design environment screen.



HDL Planning Software For AT40K FPGAs: untitled
File Edit VHDL Tools Everest Views Reports Help
Define
Instance

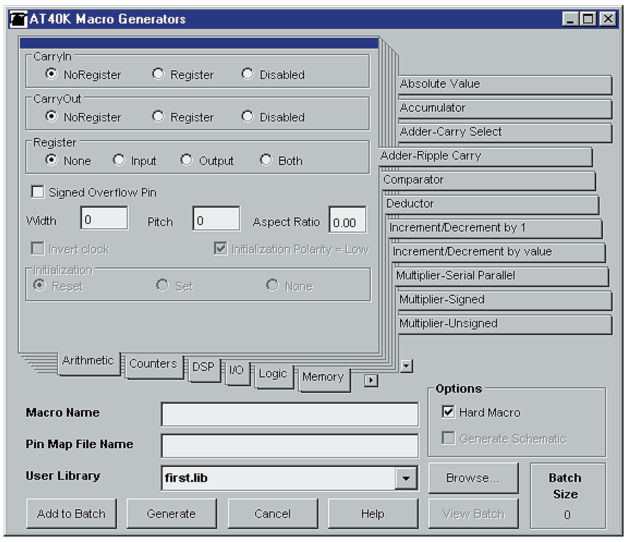
Text: Default (Fixed) Category: Register Component: latch0en-Transparent Latch, with output

```
--
-- Deleted unwanted libraries
--
LIBRARY ieee ;
USE ieee.std_logic_1164.all;
ENTITY entityName IS
    PORT (
        );
END entityName;

ARCHITECTURE behaviour of entityName IS
--
-- Add SIGNAL definitions here
--
BEGIN
--
-- Define the architecture in terms of components listed in component pull down list
--
END behaviour;
```

File: untitled Ln 1 Col 1 EDITING

Figure 6. HDL Planner screen.



CarryIn: [✓ NoRegister] Register Disabled

CarryOut: [✓ NoRegister] Register Disabled

Register: [✓ None] Input Output Both

Signed Overflow Pin

Width 0 Pitch 0 Aspect Ratio 0.00

Invert clock [✓ Initialization Polarity = Low]

Initialization: [✓ Reset] Set None

Options: [✓ Hard Macro] Generate Schematic

Macro Name

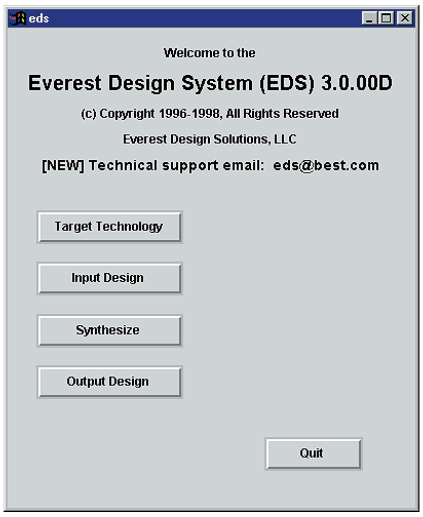
Pin Map File Name

User Library first.lib

Batch Size 0

Figure 7. Typical Macro Generator screen.

ure 8. First, the design libraries must be specified to be the AT40K libraries. Use the *Target Technology* button and selecting the *at40k.lib* option. This will force the synthesis tool into producing the most appropriate output for the AT40K FPGA architecture.
The next task is to input the design, achieved by using the *Input Design* button. This allows you to browse to the input VHDL file and specify it. If there is more than one VHDL file in a design, they may all be entered using the EDS command line that appears in the log output window. As well as reading the design into the EDS environment, this process performs a syntactical and semantic check on the VHDL code. This will highlight any areas of the input files that have constructs/sections that do not adhere to the VHDL standard. Mistakes that beginners to VHDL often make are reduced by using *HDL Planner*, as this inserts the framework in the correct syntax automatically.



Welcome to the

Everest Design System (EDS) 3.0.00D

(c) Copyright 1996-1998, All Rights Reserved

Everest Design Solutions, LLC

[NEW] Technical support email: eds@best.com

Figure 8. EDS design stages.

Once a design has been successfully read in to EDS, we can synthesise the design. There are a few options available from the *Synthesise* button, see **Figure 9**. The first option is *Automatic Pad* Insertion. If pads are not inserted, the design cannot be correctly read into the place and route routines. Pad insertion may be achieved manually, but it requires the user to map each design to a port library component in the VHDL files. Automatic pad insertion simply does this for you. At this point, the design pinout is not set, just pads associated with signals. There are several kinds of pads, found by looking within the List Ports window. So for this example you should set the Auto Pad Insertion On
Optimisation level simply specifies how hard the synthesis routines work. The higher the level of optimisation, the longer the design takes to synthesis, but the result will have a higher efficiency in general. *Execute* performs the synthesis, which takes five stages to complete.
Finally, we need to output the EDIF format file generated by the synthesis. This is achieved by using the Output Design button, and specifying the output file name.
Having synthesised our design, we can now read it into the IDS 6.00 Place and Route routines.

Open

Open reads in the design with options. We can read in the design as a Design or a Macro. A macro is a user-defined hierarchical block that needs to be written, synthesised, placed and routed as a stand alone component. For macros, pads should **not** be inserted into the design. Most of the time we will be dealing with complete designs. Select the Design button in this case. As the design is read into IDS 6.00, as with EDS, the design is checked to see if it meets the required design rules for the place and route routines to operate on. With a VHDL design entry, it is quite rare to get an error at this stage as getting the design to synthesise usually has eliminated any problems. Any errors should be covered by the on-line documentation, but any persistent problems should be dealt with by Kanda Systems. Once the design has been read in, a window called the *Design Browser* will appear. This lists all of the design instances (i.e. library components) used to implement the design.

Map

Mapping is an optional function designed to increase the efficiency of a design. It takes the original design and tries to map (or convert) it onto the AT40K architecture in a more efficient way than by synthesis alone. For instance, if a design has synthesised to having 2-input logic functions, these can be mapped into one AT40K core cell. The net result is a smaller, and often faster design. Also, mapping will remove double inversions and double buffering where it is not required. However, if a chain of buffers/inverters has been used to create an asynchronous delay, this will be mapped out of the design to at most one level of inversion/buffering, removing the delay. Leave Mapping enabled.
To enable/disable mapping, use the *Options* menu, and select *Options* (alternatively press O). Select *Mapping* from the list box, and then the *Mapping Enabled* box may be checked or unchecked. It may be of interest for the reader to look through all of the items under the *Options* dialogue, and they are all explained fully in the on-line documentation.

Parts

This window allows the user to choose which part is to be used to implement the design. IDS 6.00 supports devices from 5K to 40K equivalent FPGA gates, with between 256 and 2304 core cells. This dialogue also allows the user to add several parts,

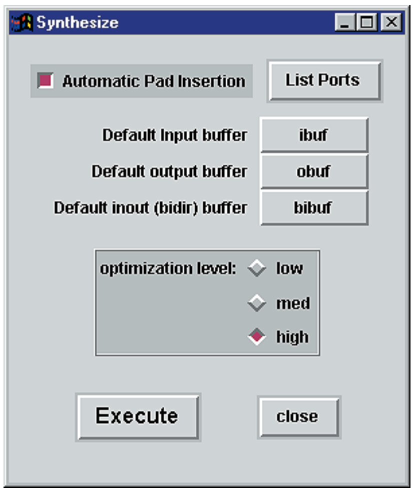


Figure 9. Options available for the Synthesis button.

and the design software to automatically place and route the design over more than one device (useful for very large designs), whilst automatically allowing for the interconnectivity between the devices.

Whilst using this window it is possible to do several functions. Firstly, we can define physical pins to the pads we inserted in our VHDL design. This allows the specification of a part ready for PCB layout, and for the Development Board peripherals to be connected to the correct input or output. The actual assigning of pin locks may be done in one of two ways.

Firstly, with the *Parts* window active, there is a command under the *Edit* menu known as *Assign Pin Locks*. This allows the user to directly specify the design pad instance to a pin number. This is very good for instance where the PCB layout has already been decided, and the device just needs to be specified with this pinout, or where the pin number is explicitly required (again, as in the Starter Kit).

Secondly, and more suited to circumstances where the FPGA design is to be specified first (usually the case when a design still has some uncertainty), the user may simply 'drag and drop' an instance of a pad onto a pin. This allows buses to be assigned in order for a port, for example. This is not always the best design method for high speed designs as a number of adjacent outputs (or inputs) all switching at the same time may cause crosstalk, but for most designs should not be a problem.

The ability of a pinout for a design to be locked prior to the final FPGA design was mentioned last month. The AT40K family has such a high level of routability it is possible to get a placed and routed design running at full speed with almost any pinout. Note that global clock signals are best used for clock type structures, as with the global reset, but above this restriction, almost any pin should be achievable.

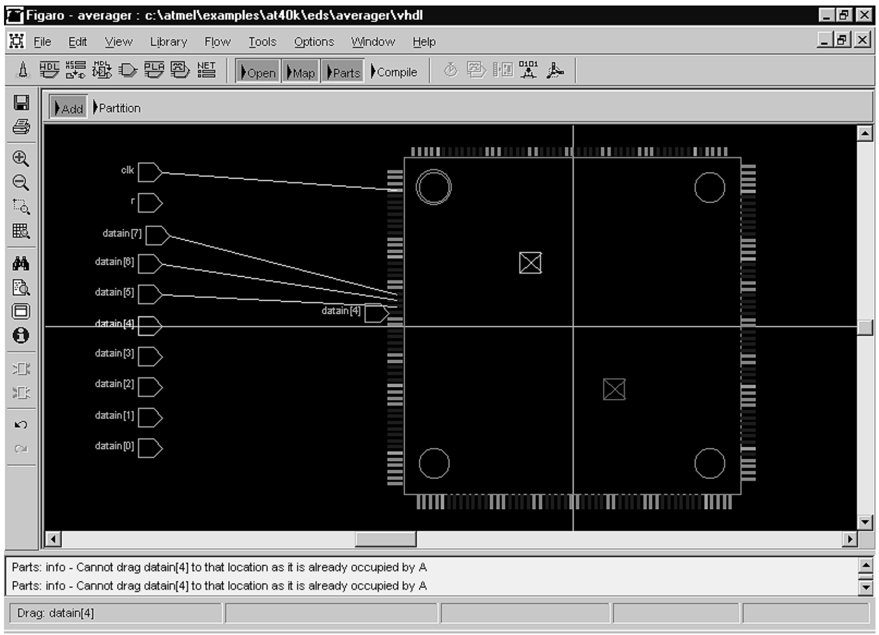
Also within the *Edit* menu from this window is a command called *IO Pad Attributes*. In last month's article we mentioned that each pad is individually assignable with pull up/downs, Schmitt trigger inputs, slew rate adjustable outputs and a number of other options. This is achieved with a VHDL design through this dialogue box. All of the design Inputs, Outputs and Bi-directional signals are listed along with their possible attributes. Each one may be specified on a pin by pin basis, so it is possible for pins in a to have different sets of pin parameters. **Figure 10** shows the *Parts* window as a Design Input is being dragged onto a pin.

The *partition* button partitions the design over a number of devices as previously mentioned, before we get onto the actual Place and Route of our design.

This button runs the *Auto Compile All* process. If, prior to clicking in this button, you select *Window – New Compile Window*, the *Compile* window appears. This will make the following explanation clearer, but is not totally necessary. This button automatically places and routes the whole design and produces the final bitstream we can use to program our device with. There are actually 5 stages to the Place and Route process. **Figure 11** shows a typical fully placed and routed design.

Firstly, we have *Initial Place*. This stage takes all of the design instances and puts them onto the silicon floorplan IDS 6.00 uses to perform the *Place and Route* operation, displayed in the *Compile* window. This is mainly a geometric exercise, keeping related design instances within a (IDS 6.00 determined) radius. At this point the design instances are allowed to 'contend' with each other (i.e., overlap). This is a single stage process allowing a first pass of placing the design.

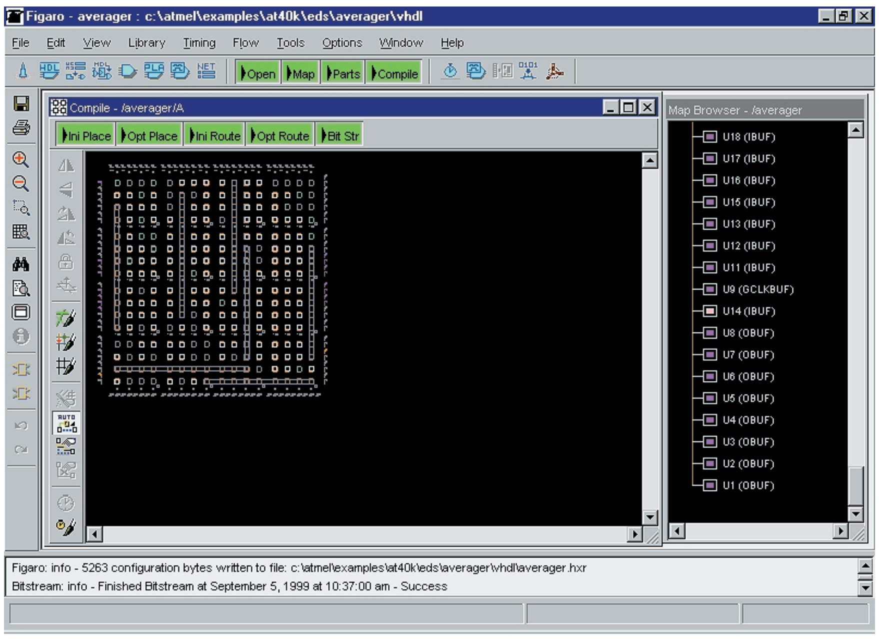
Next comes *Optimise Place*. This stage takes the placement *Initial Place* made and tries to remove all of the placement contentions. This must be done prior to routing the device, as a single cell can only perform one function in a design at any given time. This is a two-stage process, and in a complex design may take a little time. However, this may be overcome in a known simple design by using an option in the *Option* dialogue called *Place and Route*. Within this, there is a parameter called *Quality*. This represents how hard the Place and Route tools work to achieve their goal. The greater the number, the harder IDS 6.00 works, and the longer each stage takes. There is a check box called *Auto Set Parameters* which makes IDS 6.00 look at the


Figaro - averager : c:\atmel\examples\at40k\eds\averager\vhdl

Parts: info - Cannot drag datain[4] to that location as it is already occupied by A
Parts: info - Cannot drag datain[4] to that location as it is already occupied by A

Drag: datain[4]

Figure 10. This is what happens to the Parts window when a Design Input is being dragged on to a pin.



Figaro - averager : c:\atmel\examples\at40k\eds\averager\vhdl

```
Figaro: info - 5263 configuration bytes written to file: c:\atmel\examples\at40k\eds\averager\vhdl\averager.hxr
Bitstream: info - Finished Bitstream at September 5, 1999 at 10:37:00 am - Success
```

Figure 11. All done — a placed and routed design.

design and set the *Quality* level as appropriate. The quality level is used for the first four stages of the place and route operation.

Once all of the placement contention has been removed, it is possible to move onto the *Routing* of the design. But what if not all of the contention can be removed? There are three options on this. Firstly, a higher *Quality* setting may be set, and the *Placement* routines run again. Secondly, a larger device may be required. This will give more logic resource and so will usually cure the problem. With all of the available 40K devices being pin compatible within a specific package, this is possible across the range of devices from 5K to 40K equivalent gates. Finally, and for some most interestingly, we can manually modify the placement.

Manual placement allows the moving of a cell or macro around on the device to remove contention. This is done by selecting an instance on the floorplan, and dragging it to its new position. Once an instance has been selected, it turns yellow and, for a macro block, may be rotated or flipped and then moved to a suitable area. For a macro, this is possible due to the symmetrical nature of the array, and the fact that macro functions are pre-parameterised for all orientations of the macro.

Initial Route is a function not dissimilar to Initial Place, but obviously it works on the routing structure and resource. This time the cells and macros are connected together without too much regard for using the same routing resource twice. This is a three-stage process as there are several aspects of routing to be considered. As mentioned previously, the AT40K family of FPGAs was designed with routeability in mind, so this process often gives very good results prior to Optimise Place.

Optimise Route does the same for routing as *Optimise Place* does for placement. This is usually a much easier stage for IDS 6.00 software due to the routing based architecture, and if a design can be placed within a device, there will be very few times the device cannot be routed. If, however, the design does not route, what then? Well, the same applies for routing as for *Placement*. Going to a larger device will usually solve the problem. Setting a higher quality factor again will most times get rid of the contention. Finally, it is possible to manually edit both Placement (where the placement can be tweaked to allow sensible data flow), and *Routing Resource*. *Manually Routing* the device is something best left until the user becomes very confident with the devices, as what seem to be small changes can have timing impacts on a design!

The final stage for the design is the Bitstream generation. This stage generates the bitstream in several output formats, ready to be downloaded directly into the FPGA or to program a configuration Serial EEPROM, ready for programming on the user's board.

Next Time — How About Some Hardware?

We can see that the process of generating a bitstream from an initial ASCII text file comprises a number of steps, although none of them are too complex, and the whole process does become second nature after a while. Now we have seen how to generate a bitstream from a design, what about running this design in hardware? Well, using the Development Board and the 'Get Going with FPGA' book accompanying this series we will next month look at the FPGA from a hardware point of view, describing peripheral use on the board, and how to connect up a configuration device for progressing a design onto a user's own PCB.

(990063-2)

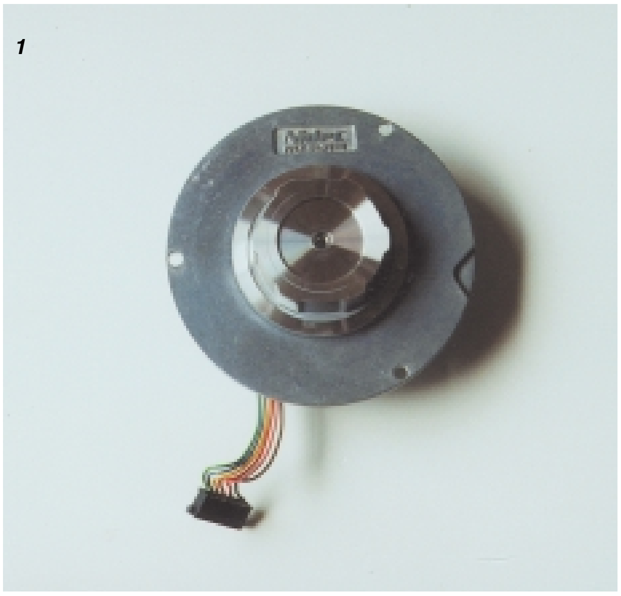
Note:

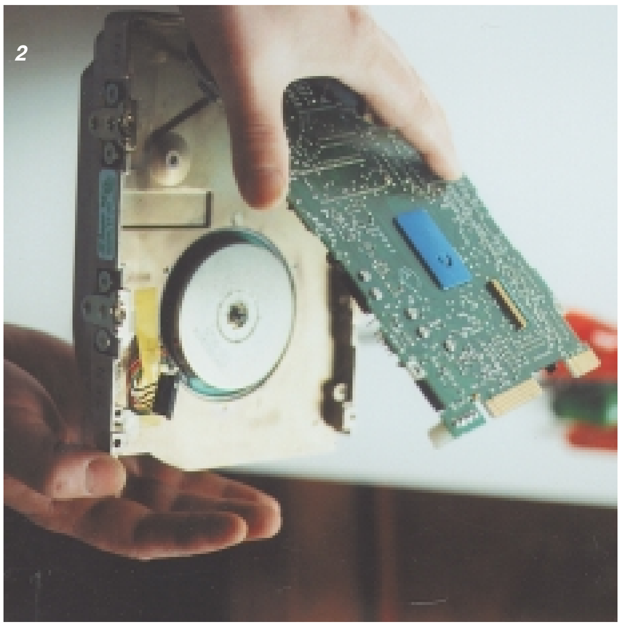
The book and Starter Kit mentioned in this article series are available (at a special discounted price for Elektor readers) from Kanda Systems Ltd, Unit 17-18, Glanyrafon Enterprise Park, Aberystwyth, Ceredigion SY23 3JQ. Tel. (01970) 621030, fax (01970) 621040, Email *sales@kanda.com*

Developments in the computing industry progress rapidly. Last year's expensive acquisitions are already no longer usable for running modern software. However, old computers, or at least parts of them, need not always end up as electronic scrap. Let's look at the motors found in old hard disk drives, for example…

By H. Neumann and E. Möller

recycling hard disk drive motors

1


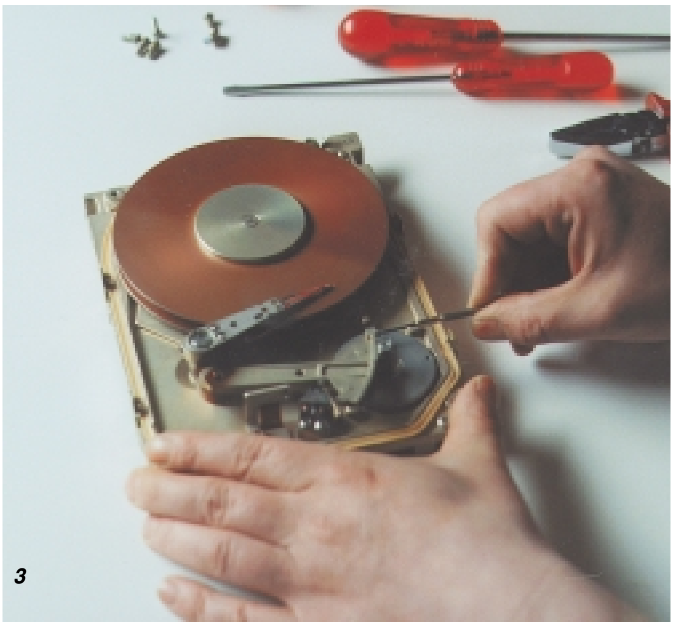
2


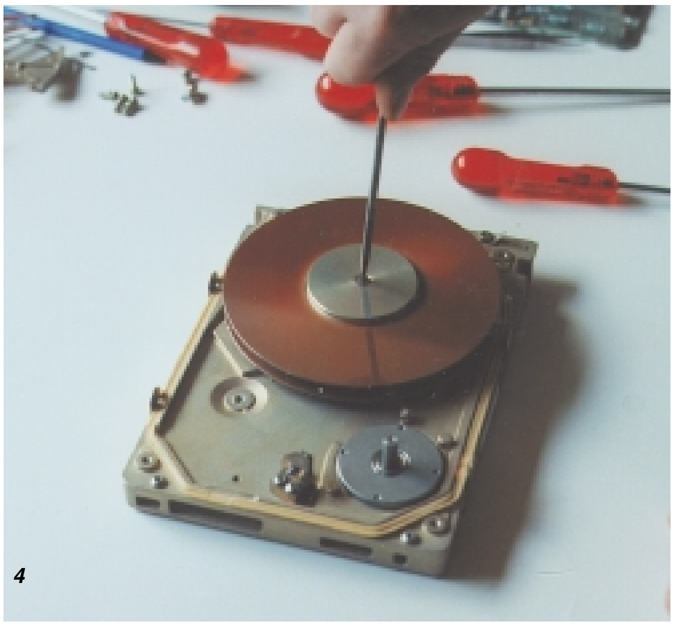
Hard disk drives with capacities of 20 to 40 MB can often be found in discarded computers (in the basement or at flea markets). The data stored on these devices is surely of no interest to anyone, but each one contains two motors, for which some useful applications might still be found. In addition to the stepper motor, which moves the read/write heads, the actual drive motor is an interesting item. In the case of the drives investigated by the authors, this is a d.c. motor with electronic commutation. An example is shown in **Figure 1**. Our objective here is to see how such a motor can be brought back to life.

Before you can do anything with the motor, you must first disassemble the drive. Start by removing the enclosure cover and the printed circuit board. The proper tool for removing the screws is a Torx screwdriver, but a sturdy slot-type screwdriver can also be pressed into service if necessary. After the cover and the board have been removed, you will see the two motors (**Figure 2**). You can now dismount the head drive motor (**Figure 3**) and the platter assembly (**Figure 4**). Finally, remove the fixing screws for the drive motor (**Figure 5**), and you can extract the desired object.

The motor has six leads, three of which are connected to its windings. These leads can be easily identified with the help of an ohmmeter. The winding resistance will measure around 3 or 6 Ω, depending on whether you measure across one or both windings. You can thus directly figure out which leads are the winding leads. The three other leads are connected to a Hall-effect sensor IC, and the measured resistances between these leads will be significantly higher. With the test circuit shown in **Figure 6**, you can easily sort out which of these leads is connected to what. In the worst case, you will have to connect each of the three leads to earth in turn while the other two are connected to + 12 V via series resistors and LEDs. Turn the rotor of the motor slowly. If the proper lead is connected to earth, one of the LEDs will be constantly on; the associated lead is then connected to the power supply terminal of the IC. The other LED will blink as the rotor is turned; its lead is thus the signal output of the IC.

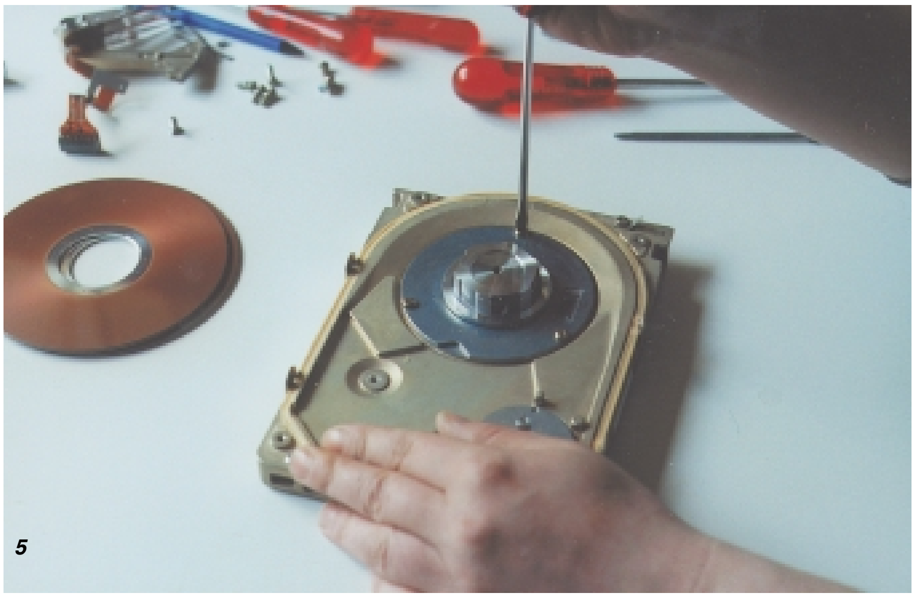
Now you have completed the necessary preparations in order to be able to use the motor. For this, you will need a circuit as shown in **Figure 7**, which can be quickly constructed on a bit of prototyping board. The Hall sensor built into the motor has an open-collector output. If the output transistor is switched on, transistor T1 is cut off. Resistors R2 and R3 then supply enough current to





switch on T2. This in turn supplies base current to T3 via R4, and T3 is also switched on. A magnetic field is created in coil L1, and this causes the permanent-magnet rotor to rotate slightly. The Hall sensor is now exposed to the opposite magnetic polarity, so that its output transistor is switched off. The base of T1 now receives current via R1, and it switches on. Since the voltage at the collector of T1 is now very low, the current flowing through R3 is not sufficient to keep T2 switched on. Consequently, T3 is also no longer switched on, and current no longer flows through L1. However, since T1 is switched on, T4 is supplied with a base current via R2, and it switches on. L2 thus receives a current and produces a magnetic field, which in turn causes the rotor to rotate a bit further. This sequence repeats itself over and over again as long as an adequate supply voltage is present. Diodes D1 and D2 protect T3 and T4 against induced voltages. If you want to reverse the motor's direction of rotation, simply exchange the L1 and L2 leads.

(992039-1)



Figures 1 through 5. How to take apart a hard disk drive.

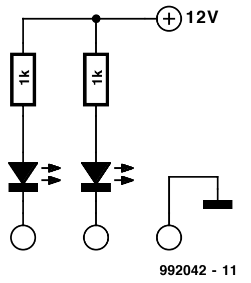


Figure 6. This test circuit can be used to determine the connection scheme of the motor.

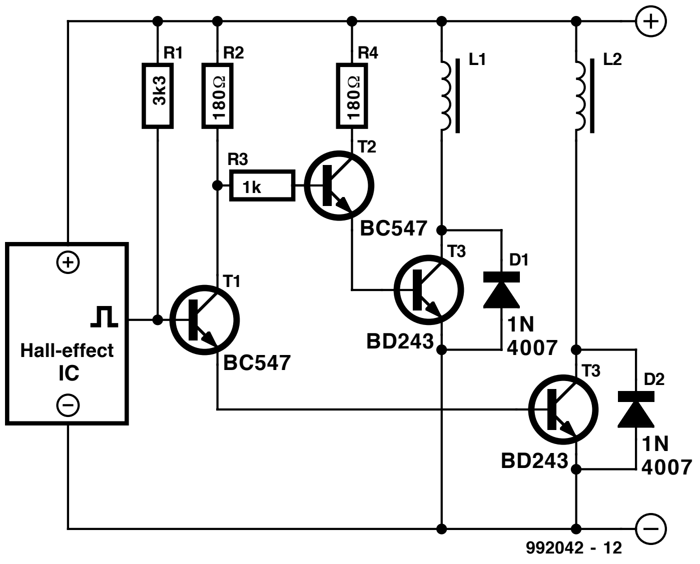


Figure 7. Control circuit for an electronically commutated motor.