elektor

73
May 1981
U.K. 60 p
U.S.A. $2

up-to-date electronics for lab and leisure

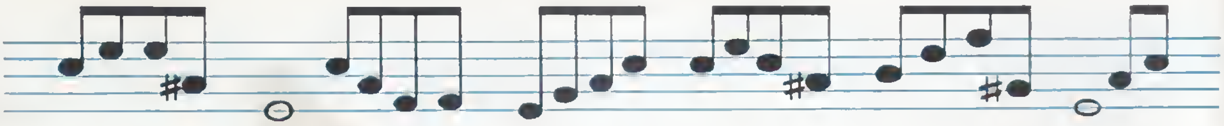
early warning
An alarm c(l)ock for campers

bar code
How prices are put in line

choke alarm
Stop pouring petrol out the exhaust

Junior interface
More memory, cassette and printer routines for the Junior Computer

Make it for a Song!



The New Maplin Matinée

Amazing Value For Only **£299.95** + £99.50 for cabinet if required.

Easy to build. Latest technology — means less cost, less components and 80% less wiring. Comparable with organs selling for up to £1,000.00. Two 49-note manuals. 13-note pedalboard. All organ voices on drawbars. Preset voices: Banjo, Accordion, Harpsichord, Piano, Percussion. Piano sustain. Sustain on both manuals, and pedalboard. Electronic rotor, fast and slow. Vibrato and Delayed vibrato. Reverb. Manual and Auto-Wah. Glide (Hawaiian Guitar Sound). Single finger chording plus memory. 30 Rhythms! 8-instrument voicing. Major, Minor and Seventh chords. Unique walking bass lines with each rhythm. Unique countermelody line with each rhythm. Truly amazing value for money.
Full construction details in Electronics & Music Maker magazine.

The complete buyers' guide to electronic components. With over 300 pages, it's a comprehensive guide to electronic components with thousands of photographs and illustrations and page after page of invaluable data. Get a copy now — it's the one catalogue you can't afford to be without.

Post this coupon now for your copy of our 1981 catalogue price £1
Please send me a copy of your 320 page catalogue. I enclose £1 (Plus 25p o&p). If I am not completely satisfied I may return the catalogue to you and have my money refunded. If you live outside the UK send £1.68 or 12 International Reply Coupons
I enclose £1.25 E5/81

Name ____________________

Address ____________________

____

MAPLIN

Maplin Electronic Supplies Ltd
All mail to: P.O. Box 3, Rayleigh, Essex SS6 8LR
Telephone: Southend (0702) 554155. Sales (0702) 552911
Shops
159-161 King Street, Hammersmith, London W6. Telephone: 01-748 0926
284 London Road, Westcliff-on-Sea, Essex. Telephone: Southend (0702) 554000
Both shops closed Mondays

Catalogue now on sale in all branches of WHSMITH Price £1.00

contents

[illegible]
21/4

EDITOR:
W. van der Horst

UK EDITORIAL STAFF
T. Day E. Rogans
P. Williams

TECHNICAL EDITORIAL STAFF
J. Barendrecht
G.H.K. Dam
P. Holmes
E. Krempelsauer
G. Nachbar
A. Nachtmann
K.S.M. Walraven

Book 2 in The Junior Computer Series

Junior Computer Book 2
This, the second in the Junior Computer Book series, follows in a logical continuation of book 1 and is an indispensible aid to users of the Junior Computer.
The book contains a detailed appraisal of the software. Three major programming tools, the monitor, an assembler and an editor are discussed in detail together with practical proposals for input output and peripherals. The complete source listing with relevant comments for all described programs, and the entire contents of the EPROM, are included.

THE JUNIOR COMPUTER BOOK 2 IS AVAILABLE NOW from

Elektor Publishers, 10 Longport, Canterbury, Kent, CT1 1PE.
An order card is enclosed in this issue for your convenience.
Price U.K. £ 4.75, overseas £ 4.95 postage and packing inclusive.

selektor

Radar for the blind

This article is mainly concerned with one technological solution to a severe handicap, total blindness; but to put things in perspective let us first consider the situation of the blind in general.
All higher animals, including man, are extremely mobile creatures. A prerequisite for this mobility is the ability to perceive the environment accurately, and in particular one's immediate surroundings within a radius of, say, 5 metres. In man and most other species, sight is far and away the primary faculty to provide this information. However, there are notable exceptions.
Bats constitute a spectacular anomaly. Most species are virtually blind and yet manage to sense their surroundings in great detail by emitting high frequency sounds (usually above the human hearing range) and by listening to the echoes received from nearby objects. Under water, dolphins and whales also employ sophisticated high frequency sound echolocation, even though they are able to see.
Blind persons are known to be able to acquire some knowledge of their surroundings by listening to noises around them, such as the sound of voices, footsteps, cane taps, etc. This, however, can only entend their mobility slightly. In the future it may be possible to provide the blind with 'synthetic vision' by coupling a small television camera to electrodes implanted in the brain. Research is in fact being carried out in this field, but developments are still at a very primitive stage.
More practical at this time is to utilise the principle of RADAR, which has been in use for more than 40 years (mainly to locate the position of ships and aircraft). When a solution was sought towards developing an aid for the blind, sound waves of ultrasonic frequency, between 20 kHz and 100 kHz, were found to be more suitable than the radio frequency normally employed in RADAR. This is mainly due to the fact that sound waves are propagated at a much lower rate than radio waves, 340 metres per second, as compared to 300 million metres per second. This ultrasonic location method is similar to that used by bats, except, of course, the echoes have to be translated into a frequency that is audible for the human ear.
A highly effective aid using the above principle is known as the 'Sonic Torch'. This was the name given to the first commercially produced device of this kind, manufactured by Ultra Electronics Ltd. . An improved version, the 'Radar-Lantern' has now been developed in Melbourne, Australia.
Weighing less than 350 grams the unit is hand held like a torch, which it strongly resembles as can be seen in figure 1. A beam of ultrasonic energy is radiated from the front of the device and is partly reflected by objects in its path. This is then detected and converted into a sound that the blind person hears in an ear piece.
The sound heard takes the form of a series of 'bleeps', each lasting 0.25 seconds and separated by an interval of silence (about 20 milliseconds). When the 'lantern' is pointed at a hard, flat surface, such as a door, or a window pane a few feet away, the 'bleeps' are heard as clear, pure tones. The pitch of the tones gives a direct indication of the distance between the lantern and the object concerned. By sweeping the instrument's 'light' either vertically or horizontally across an object, the latter's shape may be surmised. The beam is quite directional with a width of about 8 degrees, or 16 degrees in total. By combining the information obtained from scanning the beam with the pitch of the bleeps, the user can obtain a three-dimensional picture of his environment. This may range from very close (about 1 cm.) to 6 metres.
According to the surface that is reflected, the bleeps will vary from 'chirps' and 'tweets' to 'warbles' and 'swishes', etc., so that with experience the blind person can learn to 'see' every different aspect of his/her environment.
Fred Gissoni, a blind American experienced in the use of the Ultra torch mentioned above, testifies to the effectiveness of this type of aid as follows:
"Once, while demonstrating the use of the aid in a room which was totally unfamiliar to me, I was asked to walk about the room describing objects that I found. I was able to walk about finding clear aisles, chairs and doorways. Suddenly near the centre of the room I found an object which gave a most curious set of signals. Against my better judgement I described the object as giving the impression of being a table but, at the same time, having a fence in the middle of it. This description was greeted with a combination of laughter and applause. The object I was examining was a ping-pong table.
Another time, again in an area which I was visiting for the first time, I was asked to point the aid at an object and describe what I observed. After a few seconds I said that the impression I received was of a rock fence with foliage sticking through it. Upon examining the object by touch I found it to be just that: a rock fence with foliage sticking through it."
Gissoni also speaks of how he developed a "memory bank" of characteristic sounds, not necessarily restricted to the sound of the individual "beep"; for instance when standing at the foot of a flight of steps and scanning the beam upwards, the frequency of the signal increases in steps (rather like a musical scale) and this sort of thing is soon memorised.
The best way for a blind person to be trained in using the radarsonic lantern is by being accompanied by someone who can see and so can instruct him/her in the set of 16 illustrated lessons provided each lesson lasts one hour and gradually increases in difficulty. Competence is acquired very quickly; in fact the trainee can locate obstacles such as chairs, open doors and other persons, within the first quarter of an hour. After 10 hours of training, such small objects as an envelope lying on a table can be detected and picked up without any problem.
Thus, after only a few hours of practice the device can be used as a true environment sensor. The four main features of operation are as follows:

1. Lateral discernment by scanning a narrow beam.
2. Range of distance information provided by the pitch of the bleep.
3. Characteristics of the reflected surface are transmitted by varying the timbre of the signal: 'chirps', 'swishes' etc. are heard.
4. The intensity of loudness of the signal will gave an idea of the size of the object being scanned and will vary when the 'point of view' changes (when it is being viewed from the side, for instance). The signal will be at its loudest when the lantern is pointed at right angles to the surface.

As far as the lantern's technical aspects are concerned, the unit is housed in case made of plastic and with a convenient handle. Its total length is 16.5 cm and weighs 330 grammes. Power is supplied by a self-contained 9 V rechargeable NiCad battery with 150 mA hours capacity. Current drain during operation is nominally 24 mA, so that a fully charged battery will operate for 6 hours. A small battery

1

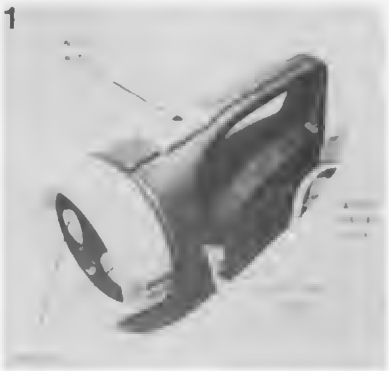


Figure 1. The Radar Lantern. This ultrasonic aid is used by blind people to scan their whereabouts and so form a three-dimensional picture.

2

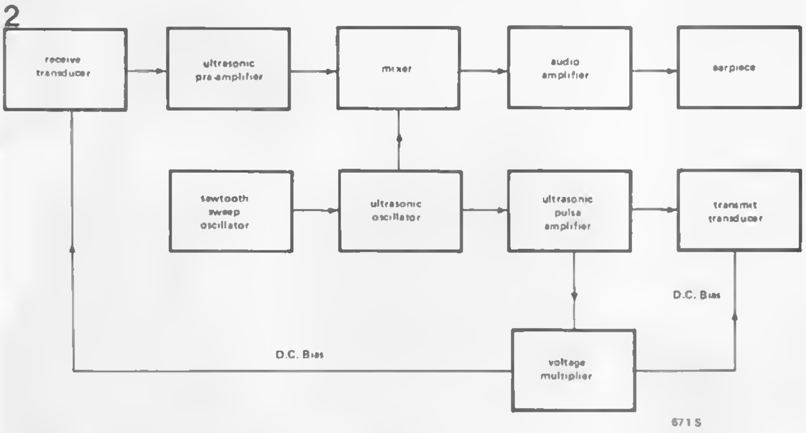


Figure 2. The block diagram of the Radar Lantern.

3

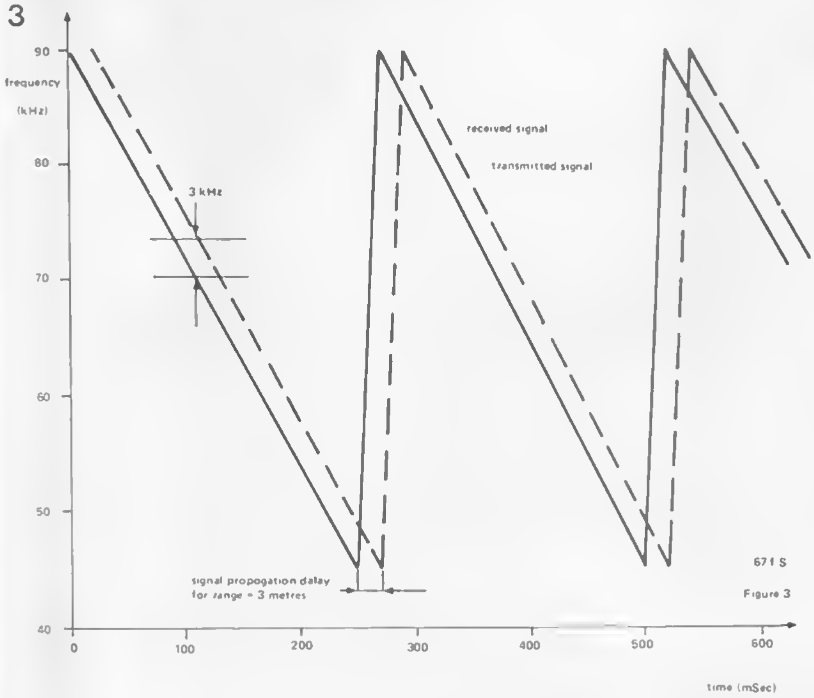


Figure 3. A graph showing the transmitted and received signals.

charger is supplied with the lantern and this will fully charge the battery in 15 hours. Of course, it will 'top up' a partially discharged battery and the charge time should last twice the duration of the period used.

Two ultrasonic transducers (one transmitting and one receiving) are housed in the front of the unit (see figure 1). These are electrostatic types, identical for transmit and receive. They require a D.C. bias voltage of 150 volts which is supplied by a six stage amplifier (see figure 2).

The electronic components include 11 bipolar transistors, 1 field effect transistor and 2 integrated circuit. These are associated resistors, capacitors etc. are mounted on two printed circuit boards contained within the rectangular portion of the case (see figure 1). The battery is housed to the rear of the P.C. boards.

Only two controls are required — an ON/OFF switch on top of the unit and a sound volume control on the right hand side. On the left hand side of the case there are two sockets, one for the earpiece and one for the battery charger so that the battery may be charged in situ. A wrist strap prevents the lantern from being accidently dropped.

The unit is basically a frequency-modulated RADAR set and the principle of operation may be explained with reference to the block diagram in figure 2 and the graph in figure 3. The sweep oscillator produces a sawtooth sweep voltage with a rise-time of 250 ms and a flyback time of 20 ms. This voltage frequency-modulates the ultrasonic oscillator causing its frequency to sweep down from 90 kHz to 45 kHz (and then return) as shown by the full line in figure 3. The ultrasonic oscillator is simply a two-stage multivibrator with "constant current" stages supplying timing currents. These "constant current" stages are controlled by the output of the sweep generator.

The dotted line in figure 3 shows the frequency of the echo signal when it is reflected from an object (a wall, for instance) 3 metres from the lantern. It will be seen that during the 250 millisecond duration of the sweep there is a frequency difference of 3 kHz between the transmitted and reflected signals. The echo signal is detected by the receive transducer, amplified and fed to the mixer (which makes use of an MPF104 FET). Since the transmitted signal is also fed to the mixer (see figure 2) the 3 kHz differential frequency will appear at the mixer output. This signal, which is in the audio range, is boosted by the audio amplifier and fed to the earpiece, thereby producing the characteristic bleep described above.

On examining figure 3 a little more closely, it becomes apparent that a variation in the reflecting object's distance would alter the time lapse between transmission and reception. Thus, the difference (audio) frequency will vary in proportion to this. Now, supposing there were two reflecting surfaces at different distances but both within the lateral width of the beam, clearly two audio frequencies would be produced and as a result, a double note would be heard.

It will be evident that in the "buzzing confusion" of the real world, full of objects both stationary and moving and of varying shapes and textures, the blind person's ears assailed by a very complex intermodulation of audio signals and the total signal heard could well be a far cry from the pure tone situation as illustrated in figure 3. Notwithstanding the difficulty of analysing such signals, however, with practice the blind person will soon learn to memorise and discriminate between the sounds heard, and this is why the timbre of the 'bleeps', as mentioned above, is so important.

The 'lantern' contains good quality components with a 0° to 70° temperature rating and the battery is designed to operate between 0 and 50°. The instrument might therefore fail to operate in sub-zero temperatures. It is however suitable in rainy conditions, although it should not be totally immersed in water.

It may be concluded that the sonic torch device offers an effective solution towards increasing the mobility and general emancipation of the blind.

With thanks to Mr. Y. W. Park and to the 'Radar for the Blind' Club, Preston (Victoria), Australia.

(671 S)

camping c(l)ock

Readers may think this is a 'cock-and-bull' story, as this electronic cock does not actually crow. It failed its oral at the crowing school, sounding more like a chirping cricket and you may think 'that's not cricket' either, for, after all, it does *look* like the farm-yard fowl — quite a handsome specimen, in fact. Nevertheless, it fulfils its task admirably and will get you up with the lark, or rather, the sun, in the morning.

Why does a camper put up his tent in the middle of lush countryside rather than Piccadilly Circus, if not with the desire to go back to nature — at least to get a little closer to the birds and the trees? What could therefore be more appropriate than to wake up at the crack of dawn to brilliant sunshine thanks to the cheerful sound of this man-made feather-brain? For it is the sun that makes it sing its dawn chorus solo, its plume being covered in a bright array of solar cells. That's natural energy for a start!
Attentive readers will now point out: "Yes, but what happens when it rains?!" Well, then there's nothing to get up for, is there?' Might as well stay in bed! And so our understanding feathered friend won't make a sound until the village pub opens. Unfortunately, however, the real dawn chorus won't be nearly as considerate, come rain or 'shine, meaning you will have to resort to cotton wool, or, if the worst comes to the worst, to the shot-gun.
Now that we've fully woken up to the idea, let us take a look at the circuit diagram of this ingenious device. This is shown in figure 1.
As you can see, this is a straightforward affair, a bit of a lark really. All it contains are a few miniscule solar cells and a buzzer. From the electronic point of view the device has nothing to be cocky about, but the main thing is, it works and it is fun to build.
A fairly conventional oscillator is constructed around T1. It is unique in that it operates from an extremely low supply voltage (0.5 V!) and provides a clearly audible alarm signal on very little current. The signal is produced by the buzzer, type PB 2720 from Toko. This is loud enough to get through to even the soundest sleepers, provided, of course, the gadget is placed in close proximity.
The camping c(l)ock's lifeline is its power supply. Ten tiny solar cells are connected in series to make sure operation is not 'half cock', even when it's pouring outside. Each has a surface area of about 1 cm^2. The types used in our prototype c(l)ock were 6 x 19 mm in size and capable of delivering 20 mA of short-circuit current. If readers are unable to obtain this particular type, any other will do, as long as the total surface area remains the same. In other words, the oscillator will work equally well, whether the 0.5 V supply is provided by two 5 cm^2 cells or by ten 1 cm^2 ones. If less sensitivity is required, the surface area may even be smaller than 10 cm^2.
The voltage divider used to preset T1 has been made variable, so that the amount of sunlight required to activate the oscillator can be preset with P1. The sensitivity has a very wide range: when P1 is set at a minimum resistance, quite a lot of sunlight is needed for the buzzer to sound; when P1 is set at a maximum level, the glow of a burning match will be enough to cause the cock to crow.

1

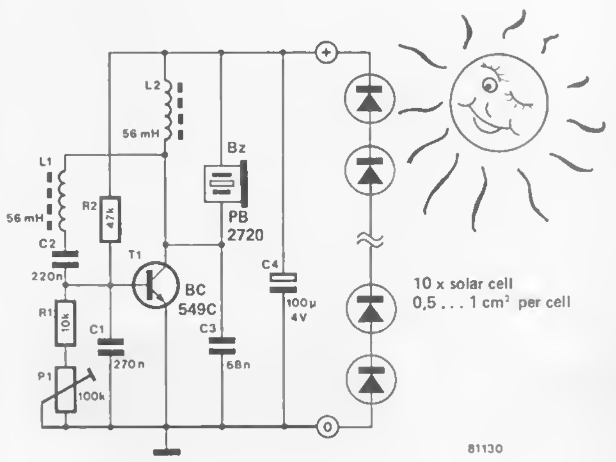


Figure 1. Considering the alarm c(l)ock is fully eutomatic end even compensates for bad weether, the circuit diagram is surprisingly straightforward.

2

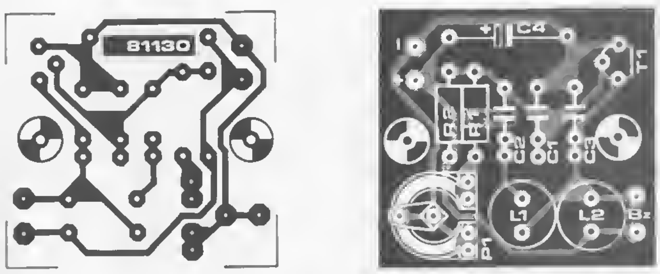


Figure 2. The camping c(l)ock printed circuit board. Construction should only teke about ten minutes.

A few practical points

Although the circuit is so straightforward that most people can probably manage to build it without a printed circuit board, our designers decided to create one anyway. The component overlay is shown in figure 2. Two holes have been drilled at equal distance from each other and from the ones belonging to the recommended Toko buzzer, thus this can easily be mounted on the board.

As far as the housing is concerned, there are various alternatives. A tiny case may be used and the solar cells can be mounted on the outside. A better solution is to insert the cells inside a transparent plastic case. It would be more practical, however, to separate the solar cells from the rest of the circuit by placing them in a waterproof perspex case with a very long cable connected to it. The solar cell case is then put outside the tent and the buzzer section is kept inside.

Of course, artistically inclined enthusiasts might like to try their hand at copying the prototype model shown on the cover of this issue.

By the way, an on/off switch is not necessary for once. When you get fed-up of hearing the cock crow, just throw a cloth over it — that'll shut him up! ◾

Parts list:

Resistors:
R1 = 10 k
R2 = 47 k
P1 = 100 k preset

Capacitors:
C1 = 270 n
C2 = 220 n
C3 = 68 n
C4 = 100 µ/6 V

Semiconductors:
T1 = BC 549C
solar cells: see text

Miscellaneous:
L1,L2 = 56 mH coil
B2 = Toko PB 2720 buzzer
Ambit

frequency doubler

Many types of effects units are available to the contempory guitarist and their use is considered the norm rather than the exception in the music of today. Of these effects, the frequency doubler is of particular interest and is fast gaining popularity with the resourceful guitarist.

Unfortunately, a good frequency doubler is not the easiest of effects units to design. This is due to the fact that the output waveforms of musical instruments bear little (or no) resemblance to those in 'the book'. In consequence, the practical result rarely conforms to the theory. The design presented here however gives a reasonable account of itself especially when cost is taken into consideration.

The basic idea of frequency doubling is essentially a good one since the resultant effect can be very desirable to the progressive musician.
By far the most commonly used method of frequency doubling is full wave rectification. This principle is relatively simple, but suffers from the drawback that it will only work effectively with perfectly sinusoidal waveforms, and then only to a certain degree. Unfortunately, unwanted harmonics are also generated in addition to the required second harmonic. Furthermore, full wave rectification will inevitably produce a DC component at the output which becomes even more of a problem as the input signal waveform approaches that of a squarewave.
It all works very well in theory, but the end results, when used with a musical instrument such as an electric guitar for instance, can be very far from satisfying to say the least! What is more, a great deal of care and attention has to be paid to the unwanted DC output component. This is of course easier said than done. Clearly, full wave rectification is a non-starter, so what other method can be used? As the frequency doubler has to be used with a guitar, a circuit was required which was capable of doubling the frequency throughout the middle range of the audio spectrum effectively, without distortion and, if possible, without any side effects. After much discussion a possible solution came to light:

$$\sin \omega t \cdot \cos \omega t = \tfrac{1}{2} \sin 2 \omega t$$

In simple terms, this formula states that the product of a sine and a cosine waveform of the same frequency is a signal of twice the frequency. It could be assumed that this will only hold good for pure sinusoidal waveforms. Wrong! The waveform of any signal can be looked upon as being made up from a (large) number of sinewaves and, therefore, the above statement will remain true.
This can be proved not only by means of formulae, but also in the form of the graph shown in figure 1. This principle would appear to be an ideal basis for our frequency doubler.

Block diagram

A simplified block diagram of the frequency doubler is shown in figure 2 to illustrate the principle of the circuit. Basically, two signals of the same frequency but which are 90° out of phase with each other, are fed to a (bipolar) mixer. Two waveforms are therefore derived from the input signal; the original waveform and the same signal with a phase shift of 90°. This gives us the sine and cosine components of the formula.
Before the signals can be mixed, the amplitude of one of them will have to be made constant, as otherwise the output signal of the mixer would not vary with the input signal, but would change quadratically. For this reason one of the signals is passed through a control amplifier to ensure that its amplitude remains constant. This is in fact called automatic gain control, or AGC for short.

Circuit diagram

Figure 3 shows how the above ideas can be put into practice. A source follower (T1) is used as an input stage in order to obtain a high input impedance. Once the signal has been split, it is phase shifted by means of two active filters constructed around a series of opamps. One signal is filtered by opamps A1 . . . A3 (and surrounding components) and the other by opamps A5 . . . A7. Opamp A4 is used to give a further 180° phase shift for the first signal so that the mixer can be controlled symmetrically.
The two signals are in fact multiplied by IC3, which is the S042P double balance mixer, ideal for this particular application. A signal of twice the frequency of the input signals can be obtained from the output (pin 3) of this IC. Switch S1 has been included so that the frequency doubler can be switched in or out of the circuit as required.
The only thing that remains to be mentioned is the AGC system. The main component in this section of the circuit is an operational transconductance amplifier (OTA). In the circuit diagram this is shown as IC4. The gain of an OTA can be varied by means of a control current at pin 5 of the IC, the larger this current, the higher the gain. The remaining components (specifically A8, D1, D2 and the FET T2) complete the AGC section.
Operation is simple: The output signal from A8 is rectified by D1 and D2. Thus, a smoothed DC signal appears at the gate of the FET (T2). The higher this DC level, the lower the voltage across R21 and the less T2 will conduct (lower drain current). Since the drain of the FET is connected to the current control input of the OTA, the gain of the OTA will be controlled automatically. This is because the control current generated by the FET will (effectively) be equal and opposite to the input signal voltage.
Frequency dependant feedback is achieved by incorporating the network R33, P3 and C15 between the output

1

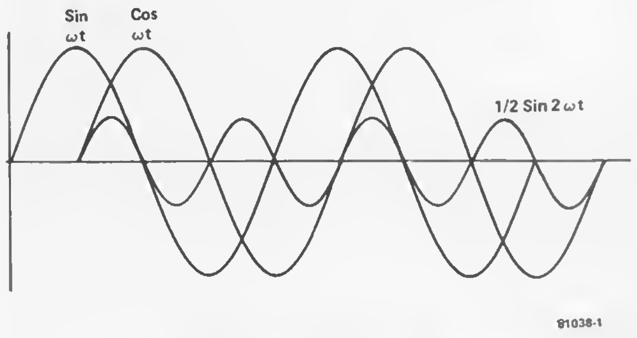


Figure 1. Sin ω t . cos ω t = ½ sin 2 ω t. The product of a sine and cosine signal of the same frequency produces a resultant signal of twice that frequency (albeit at half the amplitude).

2

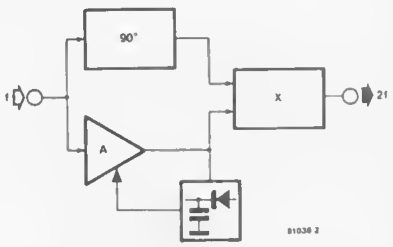


Figure 2. The block diagram of the frequency doubler. The principle shown in figure 1 can be put into effect by splitting the input signal into two and altering their phase relationship until they are 90° out of phase with each other and then mixing them.

of A8 and the inverting input of the OTA. This means that the lowest peak of the OTA can be shifted. As a result, the effect of the frequency doubler may be varied by P3 with respect to the lower frequency end of the range — depending on the characteristics of the particular OTA used. This adjustment is best carried out by ear.
Potentiometer P1 is used to adjust the control current of the OTA to ensure the correct operation of the AGC section. This will be when a quiescent current of 20 μA is measured between the drain of the FET and pin 5 of the OTA (with no input signal present). This corresponds to a voltage of 940 mV across R17. Finally, potentiometer P2 enables the amplitude of the signal 'processed' by the AGC section to be present. This potentiometer will be positioned correctly when no difference in output level is detected between the original signal and the frequency doubled signal.

Results

The protorype of the frequency doubler was found to live up to its expectations reasonably well. Its frequency range was found to be between 300 and 3500 Hz and therefore well within the spectrum produced by a guitar.

Applications for the frequency doubler need not be confined to guitar only. Enterprising misicians will no doubt be inspired to dream up all sorts of other possibilities as well. For example, tests showed that speech signals could also be doubled in frequency fairly well, although, to be honest, the audible result was far from musical! ◄

The S042P is available from Electrovalue.

3

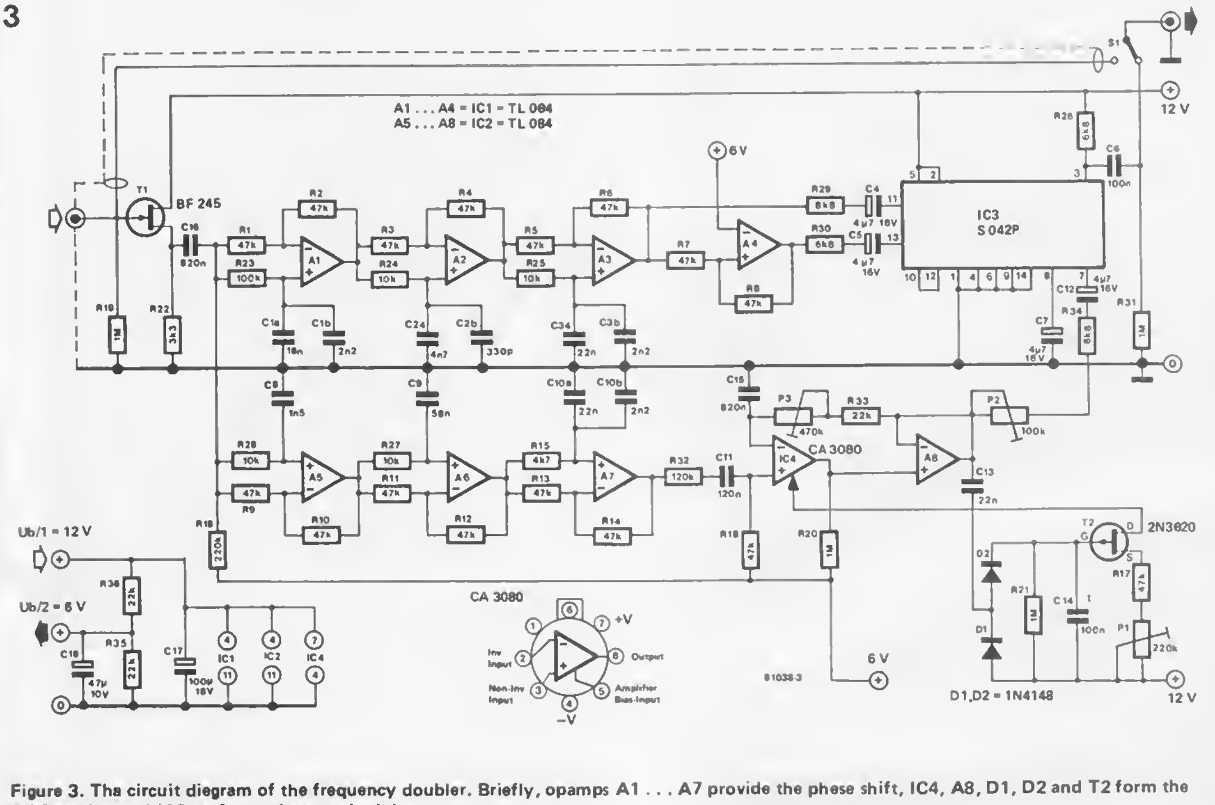


Figure 3. The circuit diagram of the frequency doubler. Briefly, opamps A1 . . . A7 provide the phase shift, IC4, A8, D1, D2 and T2 form the AGC section and IC3 performs the actual mixing.

the fully fledged Junior Computer

On board data busing provides for wider communication

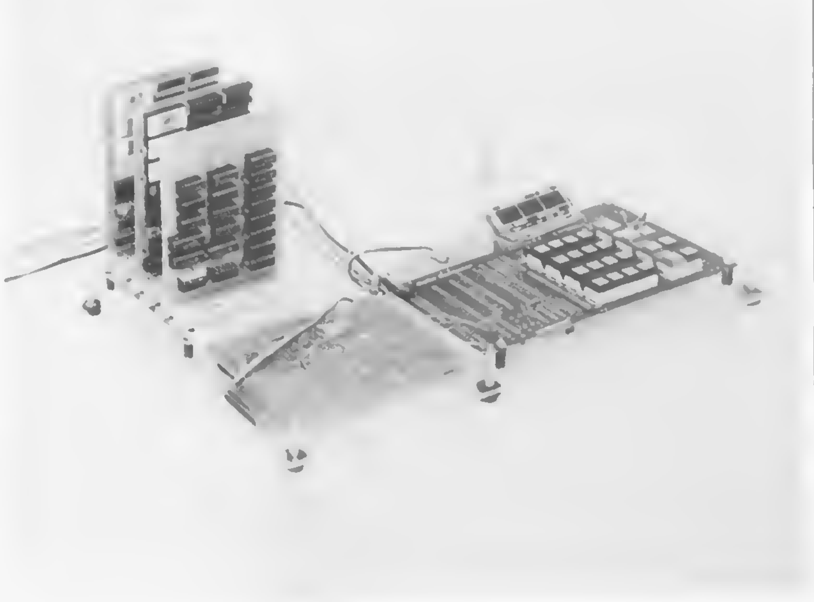
Since Junior Computer Book 2 is now available, it is time we moved on to the next and final stage in the Junior Computer project. The addition of the forthcoming interface card will transform the Junior into the complete personal computer system. This card will from the essential link between the computer and the outside world as it includes additional RAM and EPROM memory and an extra I/O system. It also enables further memory cards and a cassette recorder to be connected to its buffered bus allowing the Junior Computer to communicate with the operator (and vice versa) in a far more sophisticated manner.
This article is the first in a series of three and propose to describe the theoretical and practical aspects of the hardware extension. These articles will form the basis of Books 3 and 4, to be published later this year, and may therefore be lacking in fine detail (due to lack of space) but the intention is to give our readers a good idea of what is in store.

Now that the interface card is to be added to the Junior, the computer will be 'fully grown', that is, it will have all the hardware it requires. Its 'brain', the software, may however be further expanded. It is time for the computer to 'talk like an adult', or rather, operate in a higher level programming language and it would of course be interesting for the operator to learn to work with such a language. Elsewhere in this issue machine language and additional system software (including two software 'aids') are described in detail, but the language involved is still only machine language.
The most obvious choice would be BASIC, an 8K version on cassette, for example, combined with 16K of RAM. Naturally, this would have to be based on an existing software system such as that developed by Apple or Kim and this rather expensive option is only worth considering for the Junior Computer, provided there are enough people interested in the idea. Think about it and let us know!
In any case, Elektor is planning to provide a 16K dynamic RAM card within the near future. This would constitute an economical alternative to two RAM/EPROM cards which both include 8K of RAM.
First things first, what extensions are required to bring the Junior Computer to adulthood?

To bus or not to bus . . .

Computer systems especially designed for hobbyists are available in two types: bus systems or single board systems. The former employs several cards (printed circuit boards) to incorporate the components. The cards are all the same size (the eurocard format is 100 x 160 mm, for instance) and are individually linked by means of a 'bus'. This is a printed circuit network in which equally positioned points (such as connector pins) are interconnected. Since the system is universally compatible, the sky is the limit as far as the *theoretical* expansion possibilities are concerned, the final result being a computer with multiple facilities.
A single board system, on the other hand, incorporates everything on a single card. It all depends on what the 'everything' must comprise to suit the user's purposes (and therefore on the size of the card). Putting 'everything on a single card' is a bit of a gamble: you either win or you lose. It means the user has to estimate his/her needs very carefully from the start and this often proves to be quite a handicap, as it is difficult to plan ahead when the possibilities are largely unknown.
Single board computers are mainly used to teach operators and as part of relatively straightforward process control systems (where the computer is switched on permanently). In its standard version the Junior Computer is in fact a single board system, being designed to teach beginners. The standard version is already being used by many for all sorts of applications already, ranging from A/D conversion to process control in the manufacture of semiconductors. Thus, the interface is a purely optional addition and the expansion connector may be left untouched.
Nevertheless the expansion connector is there and for two very good reasons: it allows a cassette recorder to be connected as a backup memory (to boost a RAMshackle memory), thereby permitting full use of the total 64K memory capacity and second, it enables peripheral equipment in the way of a video terminal, etc. to be connected to the computer. This provides additional I/O and improves communication between the computer and the outside world.
Once the expansion connector is employed, the Junior Computer is no longer a single board system. This does not mean its possibilities as a bus system will be infinite. On the contrary, the computer is limited by its own hardware.

In any case, the expanded version bears no physical resemblance to the average basic system. All the extensions mentioned above are included on the interface card, which has the same size as the main board, so that they can be 'sandwiched' together. An existing SC/MP bus card may be linked to the interface card to house several other memory cards.
In spite of the bus card addition, therefore, the Junior Computer is not a bus system, but rather a 'double-decker' sandwich! However, never mind what category the computer could fall into, let's look at the system itself.



The interface card

The word 'interface' means 'link'. Here the interface card provides the Junior Computer with a vital link with the outside world in the form of various communication channels: additional I/O, a cassette interface, an RS-232-interface and an internal connection with the buffered bus board.
All the electronics involved may be found in figures 1 and 2. As can be seen, it is more elaborate than the main board, even though it is equally compact in size. Each component will now be discussed separately.

The buffers: softening electronic blows

All the connections belonging to the INPUT CONNECTOR are shown to the left of figure 1. This connector together with five links to the port connector (see figure 2) allow for data transfer to and from the standard Junior Computer. With the exception of lines EX and K1 . . . 6, which serve the interface card only, all the lines lead to the OUTPUT CONNECTOR on the right-hand side of figure 1. This connects the interface to the bus board which in turn permits one or more memory cards to be connected. The well-known A address lines are marked AB here and the D data lines are shown as DB; the B stands for 'buffered'. Why is buffering necessary? Well, for two reasons. First, it prevents an overload condition brought about by too many connections. Second, it enables data transfer along the data lines between the standard Junior Computer and the interface and memory cards to be controlled.
N1 . . . N16 act as address buffers. Since addressing always takes place by way of the 6502 on the main board, the triangles are pointing in the direction as indicated: left for inputs, right for outputs. N17 . . . N32 act as data buffers and here the even number buffer inputs point to the right and the outputs to the left, whereas the odd buffer inputs point to the left and the outputs to the right.
Three situations are feasible:

1) Ports N17 . . . N32 with odd numbers are active and the even ones are inactive when both $\overline{\text{WRITE}}$ and READ are logic 0. Data may then flow from the standard board to the extensions, if data is to be written either in the *interface* memory, the extension memory (*bus* memory) or in the I/O.
2) When $\overline{\text{WRITE}}$ and READ are both logic 1, ports with odd numbers between N17 . . . N32 are inactive (tri-state, that is to say, high impedance at both the inputs and the outputs) and the even ones are active. Data may then flow from the extension boards to the main board, if data is to be read either from the interface memory, the bus memory or from the I/O.
3) All ports between N17 and N32 are inactive when $\overline{\text{WRITE}}$ is logic 1 and READ is logic 0. No data will be transferred. This happens whenever data is being read or written inside the standard Junior memory or in the I/O. The $\overline{\text{WRITE}}$ and READ signals are generated by the PROM IC17. More about this later.

N.B. A fourth theoretical possibility is that $\overline{\text{WRITE}}$ = logic 0 and READ = logic 1, all ports in the N17 . . . N32 range being active at the same time. This should never happen.

Additional I/O

The VIA IC1, type 6522, merits individual attention. As a matter of fact, a whole chapter will be devoted to the subject in the second half of Book 3. The VIA (Versatile Interface Adapter) is remarkable in that it offers more facilities than the standard I/O nucleus, the 6532 PIA. As shown in figure 1, the VIA 'CONNECTOR' incorporates the relevant connections with the outside world. (It is put in inverted commas, as it is not a real connector). The 6522 is controlled by address lines ABØ . . . AB3 and by various signals produced by the control bus.
The eight data lines and the IRQ output signal (interrupt after a timer time out) will be familiar from Book 1.
Like the 6532, the 6522 features two chip select signals. Its $\overline{\text{CS2}}$ is connected to the output signal K6 of the address decoder IC6 on the Junior Computer's main board. The CS1 signal is linked to the output of N35 which is controlled by K6 and AB9. For the VIA to be activated $\overline{\text{CS2}}$ and CS1 will have to be Ø and 1 respectively. N35 will now act as a NOR gate, its output being logic 1 when both inputs AB9 and K6 are logic zero (in the case of the 6532, A9 has to be logic 1). Since A8 (AB8) is not connected to either the 6522 or the 6532, the VIA 6522 can be addressed as follows;
1BØØ = 19ØØ . . . 18FF = 19FF
(AB8=X:AB9=Ø:K6=1)
and the PIA 6532;
1AØØ = 1BØØ . . . 1AFF = 1BFF
(A8=X;A9=1;K6=1)
Since double addressing is excluded, as A8=AB8=X, 256 addresses are available for both the PIA and for the VIA. In the former case 19 different memory locations are available for the PIA in addition to 128 bytes of RAM. As can be seen in figure 1, address lines AB4 . . . AB7 are not connected. Thus, there are only 16 different memory locations for the VIA.

Interface memory

The interface card has room for up to 5K supplementary memory which is

1

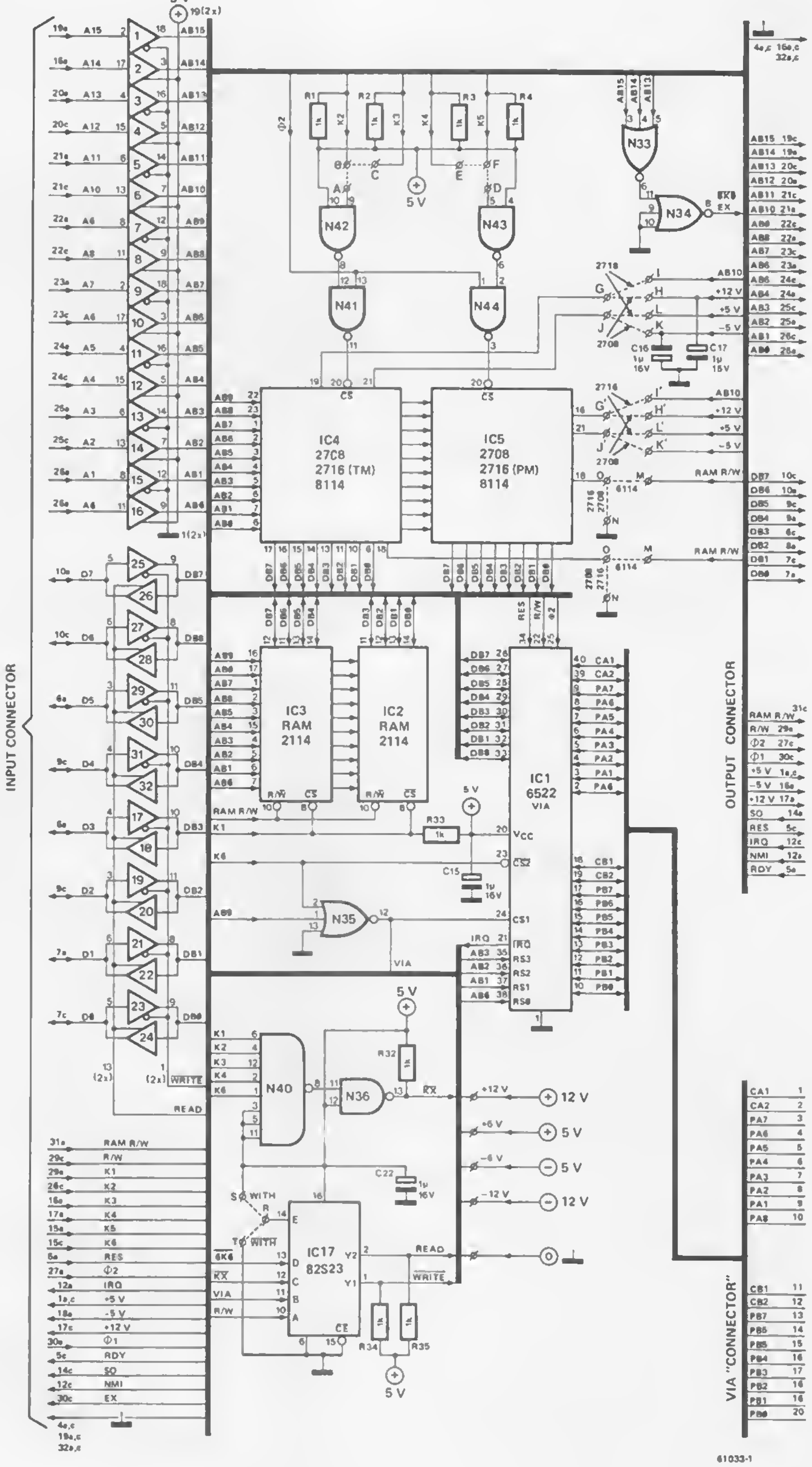


Figure 1. The interface card. It provides more memory, additional I/O, buffering including data buffer control and complete address decoding.

selected by K1 . . . K5. This is enough to allow the main board's decoded addresses (8K) to be fully utilised (including 1K for the PIA plus VIA).
In the first place, additional RAM is now available. Even without using other extension facilities, the operator can already dispose of 1K of RAM on the interface card by way of IC2 and IC3 (1024 data nibbles per IC). The RAM is selected when K1 = $\overline{CS}$ = logic Ø. Thus, its address range covers:
Ø4ØØ . . . Ø7FF.
In other words, four pages (Ø4 . . . Ø7) are available linking up nicely with the standard RAM pages ØØ . . . Ø3, so that lengthy user programs can now be stored 'in one piece'.
Both IC4 and IC5 may be 1K RAM (8114), 1K EPROM (2708) or 2K EPROM (2716) types. One or two signals K2 . . . K5 select each IC with the aid of the clock signal Φ2 (ports N41 . . . N44). The latter is necessary to time the reading and (if appropriate) the writing process.

Addressing is as follows:
K2 = logic 0 → addresses Ø8ØØ . . . ØBFF
K3 = logic 0 → addresses ØCØØ . . . ØFFF
K4 = logic 0 → addresses 1ØØØ . . . 13FF
K5 = logic 0 → addresses 14ØØ . . . 17FF

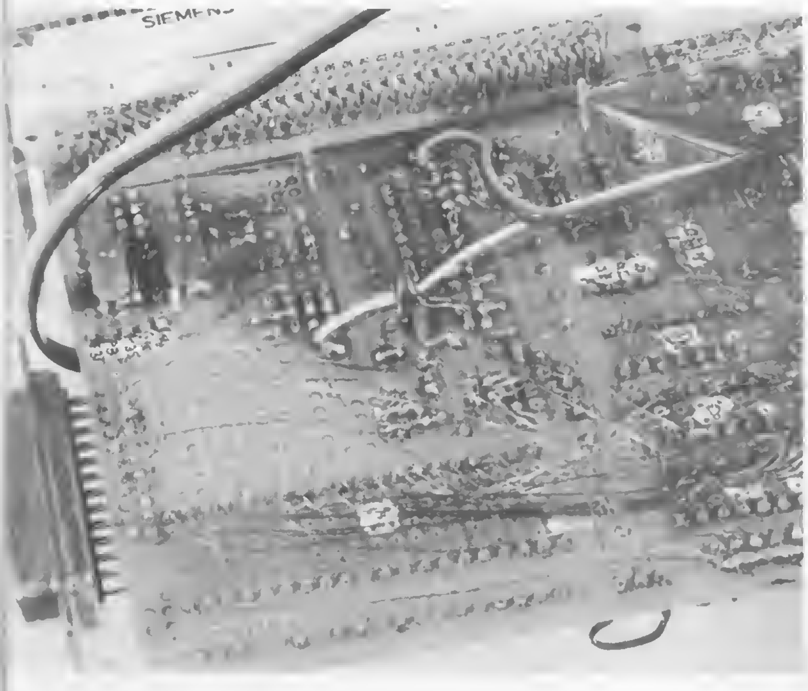
For 1K of RAM (8114) or 1K of EPROM (2708) each IC provides various selection possibilities. For 2K of EPROM (2716) two K signals are combined: K2 & K3 for IC4 (2716) (address range Ø8ØØ . . . ØFFF) and K4 & IC5 for IC5 (2716) (address range 1ØØØ . . . 17FF). Solder pins A . . . F are supplied to select these addresses. They are interconnected to suit the type of IC8. The same is true of pins G . . . O and G' . . . O', which are linked according to the value of the supply voltages and also according to whether an additional address line AB10 or the RAM-R/W signal is required or not.

Full address decoding

In the standard Junior Computer only 8K out of a total of 64K is decoded. Address lines A13 . . A15 are not used, as a result of which pages ØX, where X = Ø . . . F, are also accessed on pages 2X, 4X, 6X, 8X, AX, CX and EX, and pages 1X, where X = Ø . . . F, are also accessed on pages 3X, 5X, 7X, 9X, BX, DX and FX. In this particular situation pin D belonging to the address decoder IC6 on the Junior's main board is grounded via a wire link.
If the memory is to be extended by more than 5K, we will have to set up an address decoding system that enables us to access any address within the 2ØØØ . . . 2FFF range.
This is achieved as follows. First pin D of IC6 in the standard Junior Computer is linked to pin EX — *move the wire bridge to EX!* Pin EX=pin D is connected to the output of inverter N34 which is controlled via the NOR gate N33 and this in turn is connected with AB13 . . . AB15. As soon as one of the three address lines mentioned becomes logic 1, EX and therefore D of IC6 become logic 1 too. As a result, none of the output signals KØ . . . K7 of IC6 will be low. If all three address lines AB13 . . . AB15 are logic zero, pin D of IC6 will also be low (grounded) and one of the KØ . . . K7 signals will become logic zero. This means the I/O and memory on both the main board and the interface board can only be accessed in pages ØX and 1X, where X=Ø . . . F; pages 2X onwards can be accessed via the bus card. Supplementary address decoding is required on either the RAM/EPROM card or on the 16K dynamic RAM card to select one or more memory areas in the relevant address range 2ØØØ . . . FFFF.
The EX output signal is also sent, under the denomination $\overline{8KØ}$, to an address input of the PROM (IC17). This contains 32 bytes in all, but only two bits in each byte, Y1 (=$\overline{WRITE}$) and Y2 (=READ), are used. The two bits control data buffers N17 . . . N32. The 32 bytes can be accessed via the five address lines E . . . A, which will be described later on.
Seeing as the PROM is so complicated to access, why have one at all? Why not simply make do with the R/$\overline{W}$ signal in the manner provided on the Junior's main board? To find out, let's see what happens when data is read from the RAM, EPROM or I/O stationed on the main board. If the R/$\overline{W}$ signal is directly connected to READ and $\overline{WRITE}$ signals on the extension card, the eight data buffers with inputs facing the interface card will be enabled. These inputs will in fact not be connected to anything at all — in other words they will be left hanging in mid-air — as the interface card is not being addressed! Thus, the inputs will be in a random logic state and, as their outputs are activated, the information that appears on the data lines is bound to be incorrect. The solution is therefore to control the data bus buffers with the address system of each circuit. In the example given above (reading data into one of the ICs on the main board) the CPU 6502 does not require anything beyond the buffers and so these must be disabled. It should be noted, however, that such problems do not arise when data is being *written* on the main board.
The PROM is included for another reason as well. The NMI, RES and IRQ vectors can now (once the address decoding is complete) be stored at their true addresses FFFA . . . FFFF, instead of 1FFA . . . 1FFF (in the standard EPROM). This will only work, of course, if page FF is really stored in EPROM memory. Since page FF is included on a RAM/EPROM card here that is connected to the computer by means of a bus board, we will refer to it as bus board memory fron now on.

N.B.: As soon as a bus board is used to connect a memory card, page FF must be included on EPROM with the correct vectors, so that 'double crossing' addressing (where FF=1F, etc.) is no longer possible.

Now to get back to addressing the PROM IC17. The five address signals with which the $\overline{WRITE}$ and READ signals are obtained are as follows:
A. The R/W or R/$\overline{W}$ signal (the dash above the W merely means: 'enabled when logic zero' and helps avoid confusion).
B. The 'VIA' signal. This is obtained from N35's output and is none other than the CS1 signal inherent to the



VIA IC1. Inside the 1K zone decoded by K6=logic zero, the PIA is situated above the data buffers and the VIA below them.

C. The $\overline{KX}$ signal. This is derived from N36 which is enabled by the output of N40. This is logic zero when one of the K1 . . . K5 lines is logic zero, in other words, when the interface memory is being accessed.

D. The $\overline{8KØ}$ signal. This is another name for the EX signal which was mentioned earlier and will be logic zero when the first 8K (of memory and I/O on both the main and interface board) is being addressed and will be logic 1 when bus board memory is being addressed.

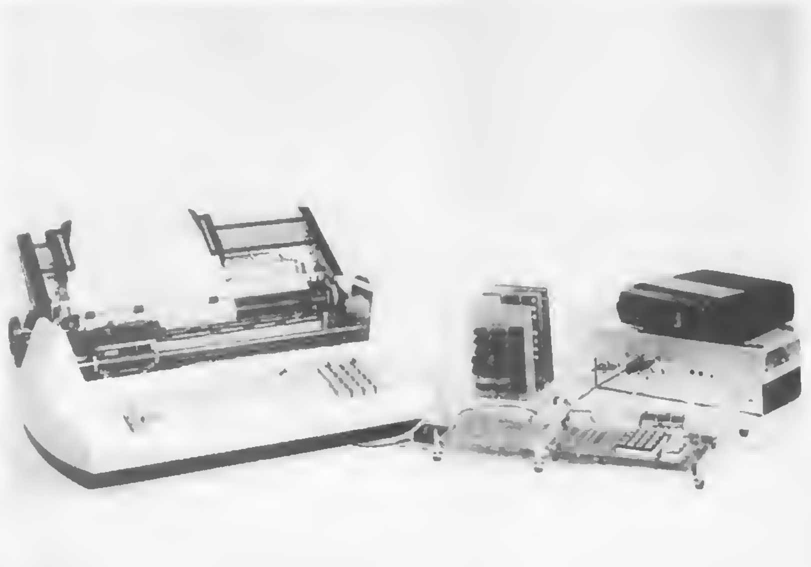
E. Pin E of IC17 is either linked up with +5 V (wire link RS) or grounded (wire link RT). Pin E *must* be logic zero when no bus board memory is connected (indicated as $\overline{\text{WITH}}$, meaning WITHOUT) and must be logic 1, when it is in fact connected (WITH, meaning $\overline{\text{WITHOUT}}$). The wire link is RS or RT is necessary for the simple reason that it enables the user to decide whether the bus board memory is to be connected or not.

Table 1 gives a survey of the contents of the PROM IC17. Bits Y3 . . . Y8 are logic zero and are not used. In principle, 32 different situations are possible leading to one of the three 'legal' READ & $\overline{\text{WRITE}}$ combinations is selected. In practice, however, only eight situations remain if the separation into read and write is disregarded:

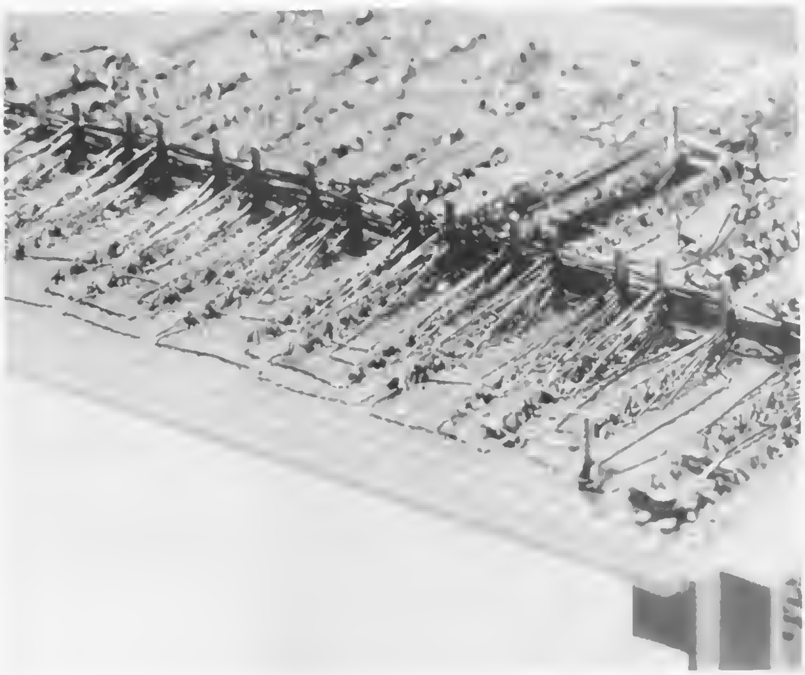
1. Writing or reading memory on the interface card (IC2 . . . IC5). Eight data buffers are enabled to indicate either read or write in PROM addresses ØØ and Ø1.
2. Reading EPROM and either reading or writing RAM or PIA on the main board. The data buffers are all disabled (PROM addresses Ø4 and Ø5).
3. Reading or writing into the VIA. Since this involves the interface card eight data buffers will be enabled to indicate either process (PROM addresses Ø6 and Ø7).
4. Writing or reading bus board memory (PROM addresses ØC and ØD). Since once again E is low ($\overline{\text{WITH}}$=WITHOUT), no bus board memory is connected and so the data buffers *must* all be disabled. The NMI, RES and IRQ vectors are automatically defined by the standard EPROM of the Junior Computer.
5. See point 1 (PROM addresses 1Ø and 11).
6. See point 2 (PROM addresses 14 and 15).
7. See point 3 (PROM addresses 16 and 17).
8. This partly corresponds to point 4 (PROM addresses 1C and 1D). As in cases 5 . . . 7, E is logic 1 (WITH=$\overline{\text{WITHOUT}}$) which means the bus board memory must be connected. Now however the data buffers will have to be enabled to allow data transfer to take place in one direction or another. The

tabel 1

PROM-address (hex)	E = WITH of $\overline{\text{WITH}}$	D = $\overline{8KØ}$	C = $\overline{KX}$	B = VIA	A = R/W	Y8	Y7	Y6	Y5	Y4	Y3	Y2 = READ	Y1 = $\overline{\text{WRITE}}$	PROM-data (hex)
ØØ	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
Ø1	Ø	Ø	Ø	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	1	1	Ø3
Ø2	Ø	Ø	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
Ø3	Ø	Ø	Ø	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
Ø4	Ø	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	Ø1
Ø5	Ø	Ø	1	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	Ø1
Ø6	Ø	Ø	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
Ø7	Ø	Ø	1	1	1	Ø	Ø	Ø	Ø	Ø	Ø	1	1	Ø3
Ø8	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
Ø9	Ø	1	Ø	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
ØA	Ø	1	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
ØB	Ø	1	Ø	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
ØC	Ø	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	Ø1
ØD	Ø	1	1	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	Ø1
ØE	Ø	1	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
ØF	Ø	1	1	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
1Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
11	1	Ø	Ø	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	1	1	Ø3
12	1	Ø	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
13	1	Ø	Ø	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
14	1	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	Ø1
15	1	Ø	1	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	Ø1
16	1	Ø	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
17	1	Ø	1	1	1	Ø	Ø	Ø	Ø	Ø	Ø	1	1	Ø3
18	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
19	1	1	Ø	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
1A	1	1	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
1B	1	1	Ø	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
1C	1	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
1D	1	1	1	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	1	1	Ø3
1E	1	1	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ
1F	1	1	1	1	1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	ØØ



three vectors are now sought in page FF. This must be connected to the EPROM which specifies the three vectors at addresses FFFA . . . FFFF.
It can be concluded that half of the 32 bytes inside the PROM IC17 are truly necessary. The other 16 logic states are irrelevant, since such combinations as D=$\overline{BKØ}$, C=$\overline{KX}$ and B=VIA just do not arise. In the other 16 cases however Y1 and Y2 are always low. This indicates that the data buffers are prepared for a write operation and so perfectly harmless.
That covers figure 1, now for figure 2 . .



The cassette interface

Most of the components shown in figure 2 refer to the cassette interface. This includes everything that is needed in the way of hardware (the software involved is dealt with elsewhere in this issue) to transfer data to and from a cassette recorder. When data is being read it is the tape that is emitting (playing back) data and when it is being written into the cassette it is recorded on tape.
The data is transferred to and from the 6502 μP by way of port line PB7 belonging to the port connector. When data is being written during the DUMP/DUMPT subroutine of the TAPE MANAGEMENT system program, PB7 functions as an output and so do PB5 and PB6. Post PB5 will then be low and PB6 high. As a result, the input of N38 with pin number 8 will also be low and so its output will be high (N38 has an open collector output which is high ohmed whenever it is logic 1). Since PB5 is logic Ø the PNP darlington T3 is connected up via R15 and so the *red* LED D5 (OUTPUT ON) lights and relay Re2 is activated. The two contacts of output J4 are interconnected. If these are connected in series with the motor of the recorder OUTPUT (remote control), the recorder can be switched on by means of software, provided that the recorder is ready to record.
Since PB6 is logic 1 during a write operation, T2 will not conduct, the green LED D4 will not light and relay Re1 will not be activated. The output of N39 leads to P2 via R20 and C14 and P2 presets the maximum recording level. J2 is the data output.

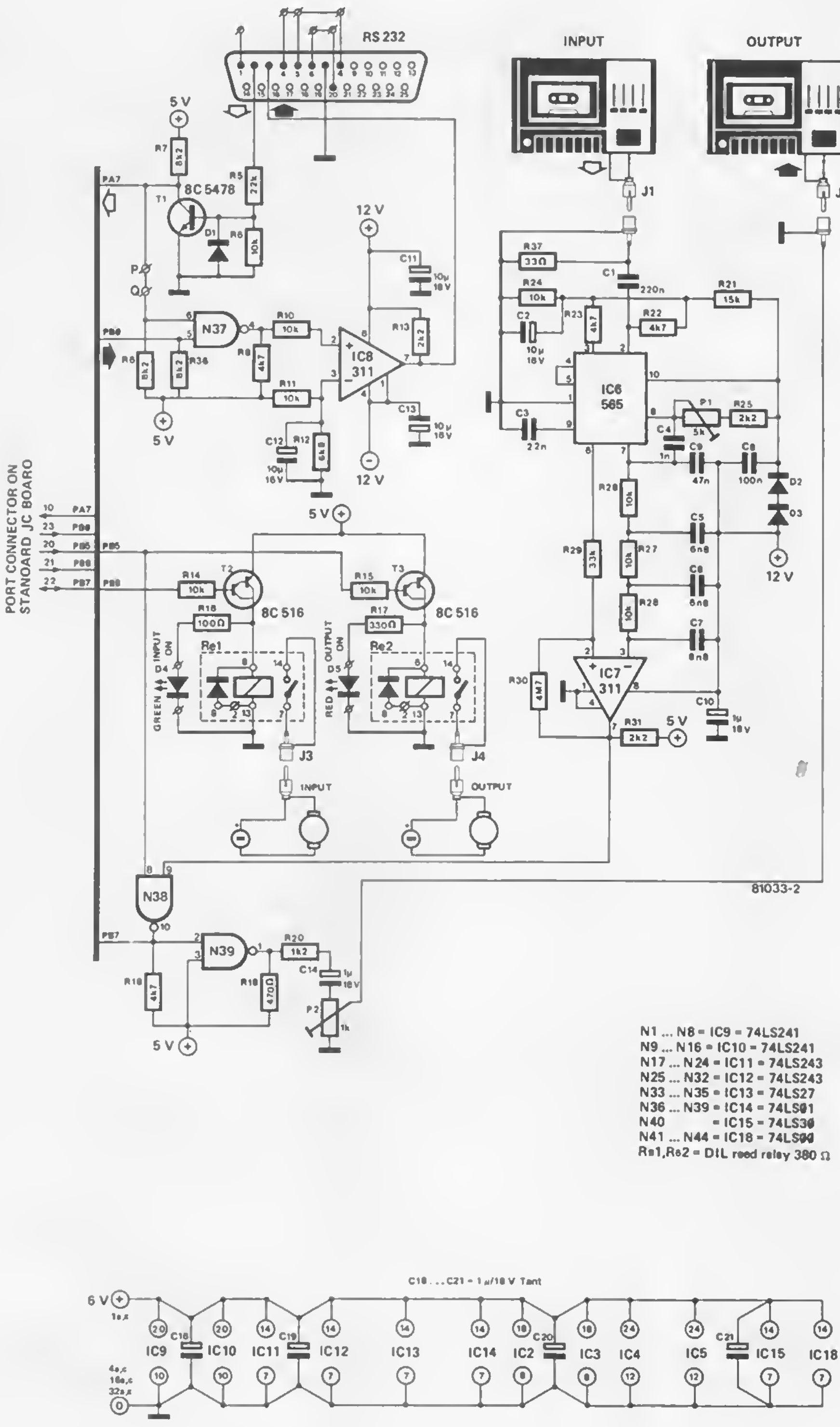
When data is read on tape during the subroutine RDTAPE in TAPE MANAGEMENT, PB7 acts as an input and PB5 and PB6 will again be outputs. Port PB5 is logic 1 now and PB6 logic Ø. As a result, N38 can pass on the output signal of IC7 to PB7 after inverting it. Port 39 remains connected through, so that signals also reach the data output J2, but without doing any harm.
Now that PB5 is logic 1, the red LED D5 will not light. PB6 will now be low causing T2 to conduct, the green LED D4 (INPUT ON) to light, relay Re1 to be activated and the contacts of J3 (remote control for the playback recorder INPUT) to be disconnected.
There is no absolute need to use separate recorders for storing and fetching data. If one is used for both purposes, remote control will no longer be necessary and the relays will then seem superfluous. However it is best to mount them *both* on the board just in case!
In addition, the T2 and T3 controls provide useful visual information about what is actually going on. The green LED lights when data is being read and the red LED is lit when data is being written.

The section in figure 2 between the data input J1 and the input of N3B (pin number 9) is next on the list. This is the circuit around IC6 and IC7 which will undoubtedly look very familiar to KIM operators. It is in fact similar to part of the KIM hardware, albeit improved, because we didn't feel like reinventing the wheel.
Before discussing IC6, IC7 and co. it is useful to know that the data is recorded on tape in the form of a series of audible signals. These are rectangular in form, a 3600 Hz signal being followed by a 2400 Hz one, and so forth. A frequency of 2400 Hz corresponds to a low level bit (logic Ø) while 3600 Hz corresponds to a high level bit (logic 1). The circuit around IC6 and IC7 makes sure the output of IC7 is high when there is a frequency of 3600 Hz at the input J1 and that it is low when the frequency at the input is 2400 Hz. The software of RDTAPE distills either a zero or a null from each cycle, depending on the 3600 Hz time in relation to the concluding 2400 Hz time. Together with the associated components the circuit around IC6 constitutes a PLL (Phase Locked Loop). To explain this in great detail (with complex formulas) would fill the pages of Elektor for a whole year! We hope readers won't mind if instead we describe its operation very briefly.

Where the interface card is involved, the PLL may be regarded as a *frequency follower*. This is because it is similar to an emitter follower, where the output voltage imitates the input voltage, an internal oscillator here adapts its frequency to that of the input signal within a certain frequency range and above a minimum level of the input signal. The internal oscillator produces a frequency that is proportional to a control voltage (VCO). Without the input signal the frequency is about 3000 Hz, so exactly half-way between the 2400 Hz. If the frequency at the input is 3600 Hz, the frequency will rise by 500 Hz; at 2400 Hz this will drop by 600 Hz. The control voltage has to be increased for the frequency to drop and lowered for it to rise. Thus, the level of the control voltage is directly related to the frequency of the input signal and after comparing it to a fixed voltage it allows a logic distinction to be made between the two frequencies: locig 1 for one frequency and logic Ø for the other. This is in accordance with the FSK (Frequency Shift Keying) principle.

The PLL is supplied with a +12 V voltage by means of D2 + D3 and C8 which are connected in parallel for transient suppression. The 'clean' 12 V (about 11 V due to the diodes) is also used to preset the input DC voltages (pins 2 and 3 of IC6) by way of R21 . . . R24. In the KIM a 5 V supply is used and any interferance will pass

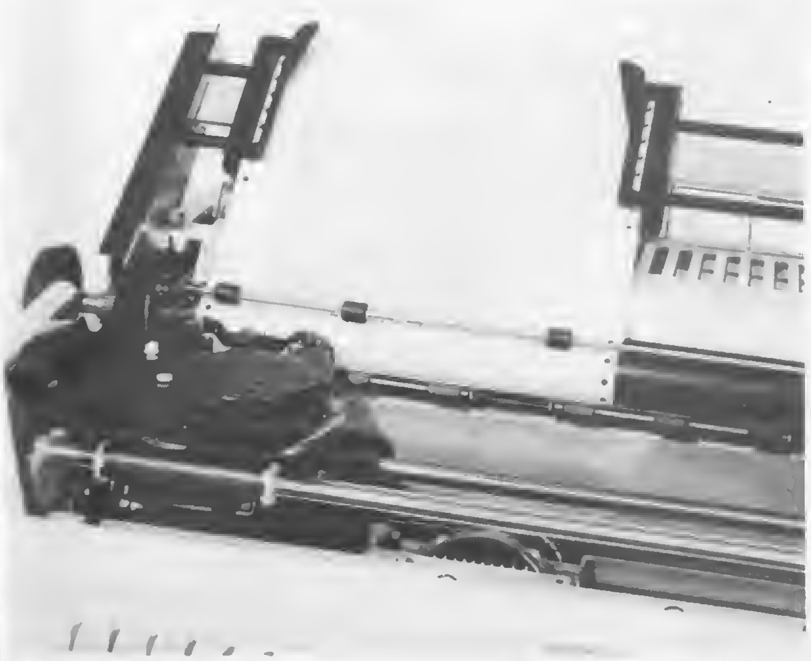
2



N1 ... N8 = IC9 = 74LS241
N9 ... N16 = IC10 = 74LS241
N17 ... N24 = IC11 = 74LS243
N25 ... N32 = IC12 = 74LS243
N33 ... N35 = IC13 = 74LS27
N36 ... N39 = IC14 = 74LS01
N40 = IC15 = 74LS30
N41 ... N44 = IC18 = 74LS00
Re1,Re2 = DIL reed relay 380 Ω

Figure 2. The circuit diagram of the interface card. Part of the circuit adapts the computer to RS 232 standards and part of it establishes communication links between the computer and two cassette recorders.

directly into the (in theory balanced) inputs. Another difference with the KIM is that the input signal is not attenuated by a factor of 10 until it reaches pin 2. All these improvements make the equipment so easy to work with that you'd have to be very careless to make a mistake when inputting data. Look out for 'drop outs' on the tape, dirty or maladjusted recording heads, etc. We happened to discover once — after a perfect data entry — that the ground connection between the computer and the recorder was missing!
Only pin 2 of IC6 (a kind of differential amplifier input) is used as an asymmetrical control. The system is connected to J1 via C1 which has a much lower value than the KIM in order to filter out the maximum of frequencies below 2400 Hz. Resistor R37 is needed in case the loudspeaker output (or headphone output) of a cassette recorder is employed.
In the absence of an input signal the VCO frequency is set by C3, R25 and P1. It is very important to preset P1 correctly, as this largely determines the quality and reliability of the data reading. Various methods will be stipulated for the correct adjustment of P1 in a later article.
Pin 7 of IC6 forms the output of the PLL. This supplies the control voltage, which was considered previously, to make sure the VCO frequency is proportional to the input signal. This pin is connected to +12 V by way of C9. Together with a 3K6 resistor inside the IC the capacitor constitutes the loop filter. This enables the PLL to react well to changes in frequency at the output without getting 'over emotional' (overshoot). By way of the ladder filter consisting of R26 . . . R28 and C5 . . . C7 the output of the PLL is connected to the negative input of the comparator IC7. Its positive input is connected to a fixed direct voltage produced by IC6 (at pin 6) via R29.
The values of the components in the ladder filter depend on the speed at which the 3.6 kHz and 2.4 kHz frequencies succeed each other. This in turn is determined by the speed with which the bits are written on tape and then read from it. This is know as the baud rate, the number of bits transmitted or received per second, and is 800 (baud) for both the Junior cassette hardware and software.
As you will remember, increasing the VCO frequency to 3.6 kHz caused the output voltage at pin 7 to drop and decreasing it to 2.4 kHz caused the voltage level to rise. Thus, since the filtered output of IC6 at the inverting input of IC7 is either higher or lower than the DC level at its non inverting input, this can be expressed as 'high' (+5 V, via R31) when the input frequency is 3.6 kHz and 'low' (practically grounded) when the input frequency is 2.4 kHz. And that is exactly what is required. The output voltage of IC7 is inverted by way of N38 ('high' goes 'low' and vice versa) and is applied to PB7.
N.B. There is a limit to the speed at which the PLL can react to a change in input frequency. Consequently, the output of IC7 will not turn over in one go from logic '1' to '0' or vice versa, but will fluctuate between the two levels before eventually making a definite choice. This is called PLL jitter and resembles contact bounce when keys are depressed. Don't worry, subroutine RDTAPE's software knows how to handle such behaviour.



Connecting peripherals by means of the RS 232C

The small circuit at the top left-hand corner of figure 2 is surprisingly straightforward, considering the enormous possibilities that it has to offer. For it enables highly complex peripheral equipment to be connected to the Junior Computer. It consists of a data transmitter and a data receiver with port PA7 acting as an input and PBØ as an output. The receiver is situated around T1 and a few other components. T1 converts logic ones at the input (top of R5) into logic zeros and vice versa.
If we suppose that input 6 of N37 is connected to +5 V by way of R8 and forget about wire link P-Q for the moment, N37 will invert the input level received from PBØ and its output will control the comparator IC8 by way of R10. A comparison is made with the

voltage at the junction R11/R12/C12. If the output of N37 is logic 1, the output of IC8 will be about +12 V and −12 V when N37 is logic Ø. Again, it can be seen that the output signal of PBØ is inverted irrespective of whether the logic levels are adapted to any particular voltage (+ and −12 V) or not. The data input (via R5) and output (IC8) are connected to a standardised 25 pin D connector, the RS 232 connector. This number refers to the universal standard that has been established for data communications. Each byte is preceded by a start bit to indicate that it has started to be transmitted. The bytes are coded according to the ASCII code.

The RS 232 standard determines the two possible logic levels and their corresponding voltage values. Later on the D connector itself was included in these standard levels. A logic zero is represented by a voltage between +5 V and +15 V (RS 232C version) and a logic one by a voltage between −5 V and −15 V. In the Junior Computer this is about + and −12 V respectively. In other words, a low voltage corresponds to a 'high' logic level and a high voltage to a 'low' level. This is known as negative logic. By inverting the logic levels in the interface twice both during transmission and reception, however, the computer does not have to bother with such subtle distinctions. The D connector features a number of pins that are interconnected. These links may be changed to suit the peripheral equipment. The first suitable device that springs to mind is the Elekterminal, a video terminal with an ASCII keyboard which was originally designed for the SC/MP, but which is equally suitable here. The system program PRINTER MONITOR is based on using the ASCII keyboard as an input (through various key commands) and the actual Elekterminal or a suitable printer (*not* — at least not for the moment — the metal foil printer published several months ago in Elektor) as an output (display).

Now all the hardware involved in the extensions has been discussed, apart from the 'revised' Junior Computer power supply and a few other modifications which need to be made to the main board. These will be dealt with in the article on the constructional details (in next month's issue) and, of course, in Book 3 which should be available soon.

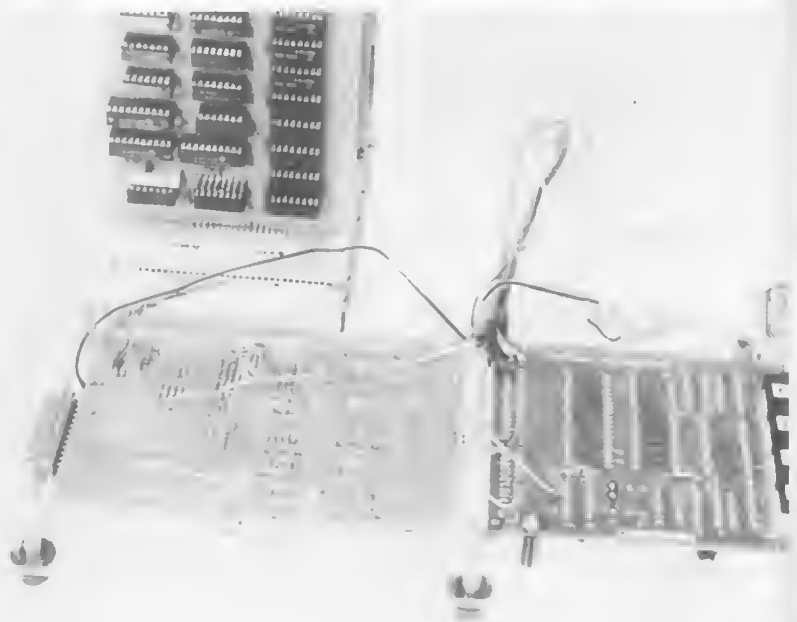
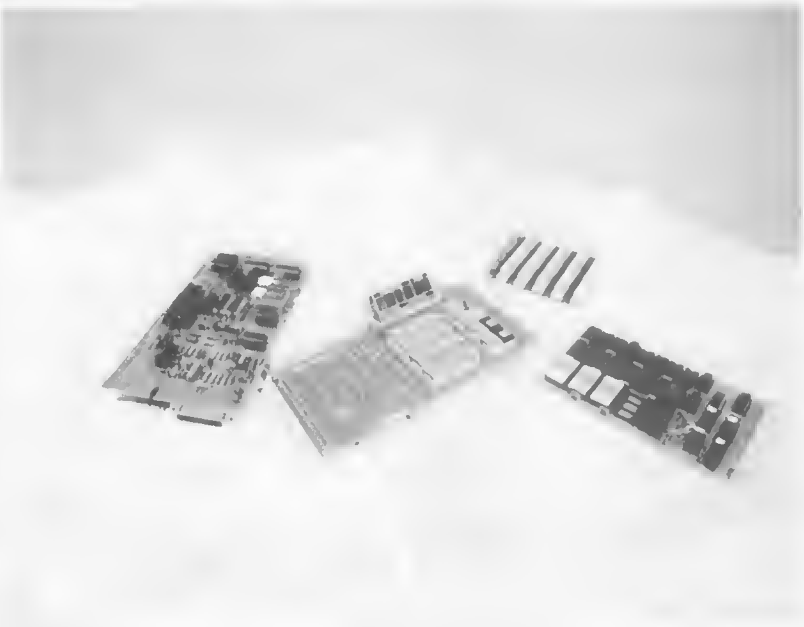


Table 1
The contents of the PROM IC7. This can be programmed by the reader with the aid of circuit no. 97 in the Summer Circuits' Issue '80 or purchased ready programmed from Technomatic Ltd. Various pin compatible PROMs are suitable, such as the 74S188, but these cannot be programmed using the above circuit. ◀



talk to computers

— and let them talk to you

Everyone takes it for granted nowadays that merely depressing a button is enough to make a washing machine do the week's washing 'at the drop of a hat', or a scanner acknowledge your credit card without the need for a telephone call to the bank. Just imagine what it would be like if computers 'evolved' yet another stage further forward and the washing machine could be activated by a mere word! Sounds like something from 'Star Wars'': talking automats that advise you on investments and give you the latest from the Stock Exchange and tell you to 'put your money where your mouth is' . . .
Seriously though, experiments are currently being carried out to equip computers with hearing and speech faculties. This would enable machines to be used for increasingly complex purposes in the future, without their being operated by electronic experts. In other words, the idea is to bring computers much closer to everyday reality.

Hans P. Baumann

According to people in the know, the first mass-produced type-writer with speech entry will be available in less than two years' time. In addition, special devices are being developed to read aloud to the blind and to teach the deaf-mute to speak and read. This article will first deal with the principles of speech synthesis and then with methods to develop a computer's 'hearing' and vocal faculties.

Once upon a time . . .

It all started in the Alps of all places, where an ancient instrument, similar to a mouth organ and often used in Alpine folk music, has been emitting vowel-like sounds (a, e, i, o and u) for donkeys' years. In contains a metal 'tongue' which vibrates within a metal frame whilst the player grips the frame between his/her teeth and shapes sounds with the hollow cavity of the mouth. Thus, the mouth becomes a resonant chamber in which the vowels are formed.
As early as 1936 K.W. Wagner hit on the idea of using this principle to construct an electrical speech synthesiser. Figure 1 shows the result in the form of a block diagram. It consists of a pulse generator that is connected to a series of formant filters. This simple device was able to imitate all the vowels and several voiced consonants (111, mmm, nnn, rrr) surprisingly well.
A few years later, H. Dudley at the Bell Telephone Laboratories managed to go one step further: based on the Wagner device he built the 'voder' (voice operated demonstrator) in 1939. This speech synthesiser could successfully imitate all the vocal sounds, provided it was skilfully operated. The voder paved the way for modern speech synthesisers and so it is well worth a closer examination. For this we must make a quick dash to the nearest telephone.
When we ring up to find out what time it is, a voice comes on the other end and drones out the time after every thirty seconds. We wonder whether it belongs to some wretched creature whose job it is to speak on tape for a full 24 hours! Don't worry, the days of such thankless labour are over. Each number and the words 'a'clock'', 'minute(s)', 'past', 'to'', 'seconds', etc. were actually only taped once and stored in a computer which is programmed to 'glean' the correct time at a particular moment from the available vocabulary.
Thus, if a machine is already capable of being programmed to 'say' separate words, it must obviously be able to pronounce the individual sounds, or phonemes, that constitute each word. For example, the word *one* is made up of three phonemes (w-Λ-n—, the bilabial 'w', the vaguely palatal Λ (uh) vowel and the nasal 'n'. Since phonemes are the smallest possible speech elements, a machine should be able to pronounce any word in a given language once it has mastered all the sounds and, with practice, it could even learn to say whole sentences, which is exactly what Dudley's voder did way back in the thirties.
The block diagram in figure 2 shows the remarkable voder. A pulse generator rich in overtones provides the raw material for voiced sounds and a noise generator produces the unvoiced ones (f, s, sh, etc.). A series of filters connected behind the generators now determine which sounds are required and imitate the formant range that is characteristic of each one. Now for an example: an 's' requires switching on a high-tone filter, whereas a 'sh' sounds a lot 'darker' and is therefore emitted by a filter with a lower frequency.
Several of the filter keys switches belonging to the voder are individually linked, so that more than one formant range can be activated at once. An auxiliary key ('soft') turns down the volume for unvoiced consonants.
Finally, there are three keys for so-called 'explosive' consonants which can be switched either to voiced (d, b, g) or to unvoiced (t, p, k). (They are called 'explosive', as each requires a slight burst of air pressure — behind the teeth in the case of d and t, behind closed lips for b and p and at the back of the throat for g and k.)
Obviously human speech is not made up of sounds alone. For one thing man is much more musical than a rusty robot and intonates while he talks. That is why a pitch pedal has been included in the device.
As figure 3 shows, the voder is such a complicated device, all things considered, that the operator has to be a real artist to be able to control it skilfully. The ideal solution is therefore to make it fully automatic, which is what most speech synthesisers are nowadays. The only other difference is their size which after being reduced to that of a Eurocard has recently shrunk to that of an IC.

Modern results

Until now speech synthesisers have belonged to two categories: word generators which store whole words in

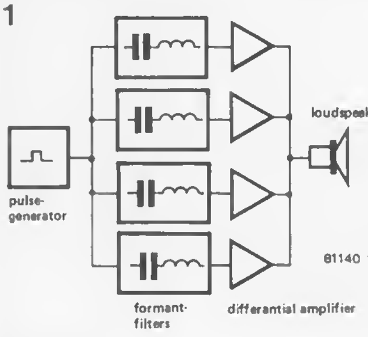
1


Figure 1. The first electronic device to produce individual speech sounds (by K.W. Wagner 1936).

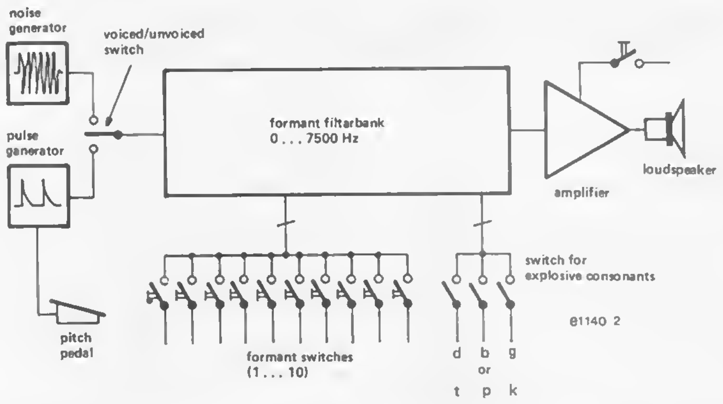
2


Figure 2. The block diagram of the voder built at the Bell Telephone Laboratories in 1939 by Homer Dudley. This device was capable of reproducing the full range of phonemes, but as it was manually operated, the result depended largely on the operator's capabilities.

memory and phoneme generators which can shape individual vocal sounds before joining them up into words.

Word generators

Storing whole words in semiconductor memories (ROMs) did not become feasible until component prices dropped and memories could be extended in a compact manner. The words are all digitalised before entry, stored in memory and then allocated to specific addresses. This enables full sentences to be formed be merely accessing the word addresses in a certain order. It has the additional advantage that anyone can program sentences without being an expert in phonetics, as they are typed in on an ordinary type-writer keyboard.

It is, however, highly uneconomical to simply digitalise speech signals by means of an A/D converter and then store the bits in ROMs or RAMs. This leads to between 50,000 and 100,000 bits of data per speech signal second! Fortunately, the human language features great 'redundancy', that is to say, even if certain elements are left out of a sentence, the remainder will be enough to carry the 'gist' of the idea across. Whether one person yells at another over a great distance, or speaks with his mouth full, or on a faint telephone line, it is surprising how much the listener is able to understand.

The word generator principle does have certain disadvantages:

1. The covabulary cannot be extended 'per program'. Although ROMs can be provided with a specific vocabulary, any modification or extension to it will mean having to manufacture an entirely new ROM. Thus, when the device is confronted with anything beyond its scope, it will be totally 'speechless'.
2. The total available vocabulary is very limited, about 250 words at the moment. Even for this small amount National Semiconductor's 'Speech Processor Set' needs 128 kbit and that is reducing memory space to an absolute minimum using all the methods described above.
3. Any sentence is highly complex in structure. It is not just a string of correctly articulated sounds, but it also involves varying the intonation and pitch, breathing spaces and emphasising certain words or sounds. These parameters are not fixed in any way, but depend on the context within which the individual words are placed. A question, for instance, sounds very different to a statement, even though the question mark cannot be 'seen'! A child soon learns to lower the pitch of his/her voice at the end of a sentence. This cannot be done by a word generator, however hard it tries, because each word is stored as a totally neutral entity in the ROM, so that the machine is bound to sound like a robot when it 'speaks'.

Figure 3. The voder (Voice Operation Demonstrator) 'keyboard'. The manual controls a series of formant filters and the operator has to be a talented 'performer' for the sounds to be coherent.

Phoneme generators

This type of speech synthesiser is largely based on Dudley's voder, as it is built up on a series of voiced/unvoiced tone generators, formant filters and pitch controllers. In addition, however, phoneme generators contain memory locations storing algorithms to produce each individual phoneme. When an instruction to form a particular phoneme appears at the input of the speech block, all the parameters are set to transform the generator signal into the required sound. Depending on how complex the sound is, up to ten parameters may be needed simultaneously (pitch, intonation, duration, 1-4 resonance filters, voiced/unvoiced, etc.).

The machine may be programmed with a standard vocabulary, and then an additional memory includes the individual phoneme codes used for each word.

Its great advantage is that it can produce any sound in a language family and so, theoretically, it should be able to master the entire vocabulary of all those languages. The Anglo-Saxon language group consists of 64 phonemes altogether — that is all that is needed to reproduce every word in both English and German. An example of a phoneme generator is the VOTRAX VS-6.0 which organises each phoneme command in the form of an 8 bit data word. 6 bits serve to select the required phoneme and 2 bits produce the right pitch, so that the utterance sounds fairly natural. The 64 phoneme codes also contain various stages in intonation and speed for every sound. The phoneme commands for the VOTRAX SC01 speech synthesiser are shown in table 1. A sound may vary in length from 250 ms for the longest phoneme to 47 ms for the shortest.

Instead of using different resonant filters as the sentence progresses, the intonation may also be varied by 15 filters connected in parallel with preset frequencies for each one. The voiced/unvoiced tone signal produced by the tone oscillator or noise generator

1

Photo 1. The VSM 2032 speech module from General Instrument incorporates a full-fledged speech synthesis system on a compact card. It consists of an interface, a speech generator IC, the SP 0250, a ROM and an NF amplifier.

2

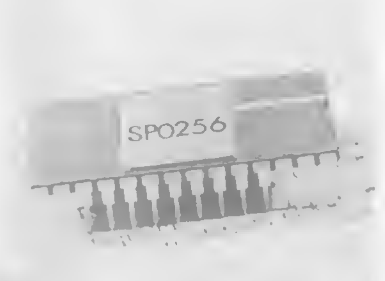


Photo 2. The SP 0256 from General Instrument takes speech synthesis one step further than its SP 0250 counterpart (see photo 1). Again, it consists of a generetor that includes a word memory, only now it also includes a 16 K mask programmed ROM. In addition the IC is able to address up to 491 K of external ROM.

respectively acts as the input signal.
Behind each filter there is a voltage controlled amplifier (VCA) which enables the sounds to change in nature and determines the contribution of every frequency range to the output signal, in the same way as in a vocoder.
The speech synthesiser also features Linear Predictive Coding (LPC) which ensures great linguistic flexibility and an unlimited vocabulary.
Combining phoneme generators and word generators reduces the need for a huge memory capacity and enables 'allophones' (transitions between phonemes) to be produced, which makes the machine sound more natural.
Phoneme generators are a little more difficult to program than word synthesisers, as each sound has to be phonetically transcribed before entry and there is a great discrepancy between the written (orthographical) and the spoken language. Thus, the operator must be up to date with phonetics and an ordinary alphabetical keyboard is no longer valid. In America this problem has been solved by developing special programs which phonetically transcribe typed-in language. This means that constantly changing information such as Stock Exchange reports, railway timetables, road reports, etc. is now much easier to reproduce.

Which system is likely to survive?

Let us compare word and phoneme generators and see which type is likely to stand up to the test of time.

Word generators

signal digitalisation using partial data reduction	± 2000 bps (bits per second)
signal digitalisation using all reduction techniques	± 600-800 bps

Phoneme generators

Linear-Predictive-Coding	± 1200 bps
Phoneme generators such as the voder	70 bps

As you can see, the voder type of phoneme generator requires very little memory capacity, as it features high speech redundancy. Since chips are getting cheaper all the time, however, word generators will probably gain popularity once their vocabulary can be increased to over 500 words despite their metallic sound. ◼

Sources:
Pedro the Voder, a machine that talks. Bell lab. Rec. 17 (1939)
The Voder. Electrician, London 123 (1939)
Rabiner, L.R. Computer Synthesis of Speech by Concatenation of Formant-Coded Words. Bell System Technical Journal (1971)
McIlroy, M. Douglas. Synthetic English Speech by Rule. Bell Telephone Laboratories Inc, Murray Hill, NJ
Teja, E. Voice Input and Output. EDN November (1979)

News Releases from:
Computalker Consultants Box 1951, Santa Monica CA 90406
Telesensory Systems 3408, Hillview Ave, Palo Alto CA 94304
Votrax Div of Federal Screw Works 500, Stephenson Hwy, Troy Mi 48084
National Semiconductor Corp, 2900, Semiconductor Dr, Santa Clara CA 9505

choke alarm

A sound economy device

Driving is a hazardous occupation on today's roads and it is not surprising therefore that a minor thing like turning the choke off is easily forgotten. This not only adds enormously to the fuel bill but also seriously increases wear in the engine.
The circuit described here is designed to attract the driver's attention to the choke by sounding an audible warning after a certain preset time period.

J.F. Verrij

In the continuous efforts by manufacturers to make driving as easy as possible many of the control functions of the car are becoming automated. The choke did not escape the march of progress and for a time the fully automatic choke became a selling point. Unfortunately, they quickly gained a reputation, rightly or wrongly, of being both temperamental and uneconomical. The oil crisis finaly put paid to them on all but the most expensive carriages and brought a welcome return of the plain and simple push-pull knob on the facia.
However, this brings us back to the original problem. Leaving the choke 'on' for too long is expensive and unfortunately, this is exactly what so often happens. Hardly anyone ever forgets to pull the choke out when starting a cold engine (a simple matter of necessity), but it takes several miles — and gallons — to remember to push it back in. An indicator light is not a lot of help either since it tends to 'grow on you' with the result that it goes unnoticed.
Still, with a little help from Elektor, this problem can be solved but before we go into that it will be as well to have an understanding of what the choke control actually does. When an engine is cold it requires a much heavier fuel to air mixture in order to maintain ignition. There are almost as many ways of achieving this as there are carburettor manufacturers. By far the commonest method is to use a 'butterfly' valve situated in the air supply passage to the carburettor. This is designed to shut off a high proportion of the air supply when it is closed — when the choke is 'on'. Other techniques are used of course, such as the sliding jet in the S.U. carburettor, but they all have one thing in common: a manual operation requiring a knob to be pulled out and pushed in with the consequent human factor.
It will be readily apparent that with the choke in operation a significantly greater amount of fuel is being used. This is detrimental in two ways. Obviously, cost is the most hard hitting point since, with the choke 'on', your carefully nursed 48 miles per gallon plummets alarmingly down to something in the order of 20 or even less. Secondly, and far less obvious, engine wear is greatly increased due to the fact that the almost neat petrol mixture in the cylinders reduces the lubricating abilities of the oil in the bores and valve guides.
It follows then that the use of the choke — a 'necessary evil' — should be limited to as short a period of time as possible. An alarm designed to attract driver's attention to the fact that the choke is still operating after a certain

1

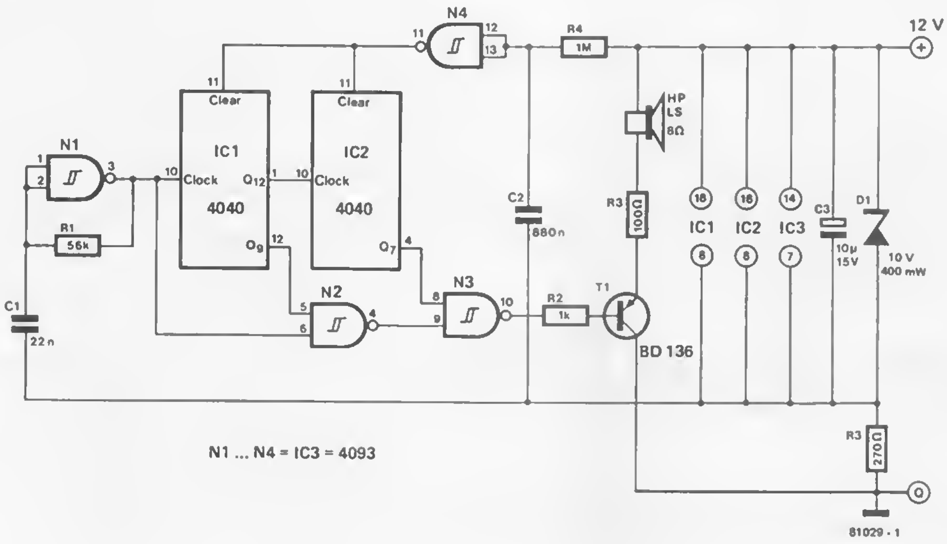


Figure 1. The circuit diagram of the choke alarm. The completed circuit is small enough to be mounted behind any dashboard.

period of time would more than pay for itself after ten or so cold starts . . . approximately one week.

The circuit here will emit a piercing squeal if the choke remains on for longer than four minutes. It also includes a choke 'on' warning lamp as an additional reminder.

The circuit diagram

The choke alarm is fully automatic and should cost no more than a gallon and a half of petrol (whatever that is by the time this gets into print). The circuit diagram is shown in figure 1 and, as can be seen, is far from complicated. The gate N1 together with R1 and C1 forms a square wave oscillator with a frequency of about 1 kHz. This is the clock frequency for the 12 stage divider IC1. The Q12 output of this IC is fed directly to the clock input of a second divider, IC2. The frequency produced at the Q7 output of this divider is:

$$1\ \text{kHz} \div 2^{(12+7)} = 0.0019\ \text{Hz}.$$

The clock frequency is also fed to one input of gate N2. The other input is connected to the Q9 output of IC1 (a frequency of 2 Hz). When the circuit is initially switched on, by pulling out the choke control knob, it will take 262 seconds for the Q7 output of IC2 about 262 seconds to change to a logic 1. This high level will, via N3, gate the output of N2 through to transistor T1 with the result that the speaker will emit a pulsed 1 kHz tone. This sound will continue until the choke (in other words the circuit's supply voltage) is

2

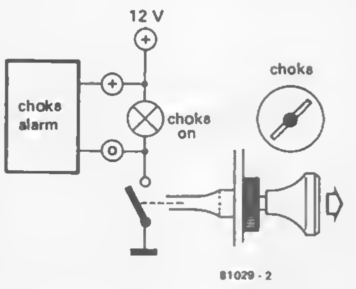


Figure 2. The choke alarm connection diagram.

switched off.

Some readers may prefer to replace the loudspeaker with a lamp that will start to flash after the time period if the noise proves to be too startling. This type of warning system, however, has been found to be far less effective in practice.

The components N4, R4 and C2 serve the rather simple but important task of ensuring that both the dividers are reset — started at zero — when the choke is first operated. This action will switch the supply on causing the output of N4 to go high resetting both ICs 1 and 2. After about 0.6 of a second, the time constant of R4 and C2, the output of N4 will revert to low and allow the timing sequence to start.

The supply voltage to the circuit is maintained at 10 V by means of the zener diode D1, resistor R5 and capacitor C3. This will prevent any interference in the car's electrical system from reaching the alarm circuit. The collector of T1, however, must be connected to the negative supply of the car.

Fitting the alarm to the car will be a simple matter if the car already has a choke warning indicator lamp and switch. The details of how this is to be carried out is shown in figure 2.

If the car is not blessed with a switch then one will have to be fitted (together with a warning light if so desired). How this is done will of course differ from car to car but a little ingenuity will find a way. As a starting point, many readers will be aware that a number of cars fitted with a warning lamp used a simple switch built into the choke cable. Since, with a little care, a choke cable can be easily shortened if necessary, one choke cable is much the same as any other . . . Try the Triumph range!

If a timing other than 4.5 minutes is required, a different output of IC2 may be chosen. For instance, if the Q6 output is connected to N3, the choke time will be 2.25 minutes. If output Q8 is used the time will be 9 minutes. Critical adjustments to timing can be carried out by changing the value of R1 although this will also alter the pitch of the warning tone. In practice it will not be necessary to get the tolerance of the time period down to this extreme. ◄

bar codes

putting data behind bars

Hard luck! A 'bar code' is not the key to the door of a public house, or even the code of conduct to follow with lady drinkers. In fact you've probably noticed it in the form of a new-fangled price tag covered in a strange pattern of black bars and white spaces. Most modern supermarkets use them, as it saves the cashier the trouble of looking up and typing the price of each item. A can of beans, along with its bar code, is merely brushed against a sensor and Hey Presto! The price appears on the cashdesk display.
But have you ever wondered what is 'behind' those bars and blanks, what could be the point of such patterns and what other uses they could have? If so, read on and find out!

Everyone knows by now that computers work in binary, that is to say, all figures are converted into ones and zeros. Looking at the bar-coded price tags, people might be inclined to think the wide bars stand for ones and the narrow ones for zeros, or vice versa, or even that there is some connection between the bar pattern and the number printed at the side. In other words, bar codes look simple enough to decipher. However, in practice it isn't that easy. In fact, numbers do not seem to have any bearing on the pattern whatsoever, as the same number may come with a variety of bar codes.

A closer look reveals that the bars are spaced very irregularly. Perhaps that could mean something too?
When various price tags are compared, the mind really starts to boggle, for even the number of bars may vary between one manufacture and another. Actually, there is a very simple explanation for this: these are several different types of bar codes.

Continuous or discrete

Every bar code is based on individual characters. Each digit has its own code — it is not converted into binary first (e.g. 85=01010101) before being 'translated' into bars.
Until now only the bars have been referred to as information carriers. Often this is extended to the spaces between them as well. If in the simplest case there are two 'sizes', wide and narrow, then bars and spaces can both be said to be either wide or narrow.
A 'continuous' code is one where *all* the bars and spaces contain information. In a 'discrete' code, on the other hand, the space between two characters is not used. Each character (either a letter or a digit) will then be represented as a certain number of bars and the spaces between them. The interval between two such groups will vary in width.
Matters are further complicated when a continuous code is used with an odd number of bars and spaces per character. Supposing the five bars and spaces determine a character. The first unit will then be represented, for instance, as three lines and two spaces, the next as three spaces and two lines, and so on.
That's not all, either. We have been presuming up until now that bars and spaces come in only two sizes. Wrong!
In a 'multi level' code various sizes are used: a standard width may be doubled tripled or even quadrupled. The advantage of this is that larger quantities of information may be squeezed into a small surface area; the disadvantage being that it now becomes increasingly difficult to print the pattern in a reliable, error-free and legible manner.

Commonly used codes

There is not much point in going through each possible code in detail. Instead, we'll select a few examples ranging from straightforward to highly complex.
The simplest variation is the '2 out of 5' bar code. This uses five bars to represent a single digit. Two out of five bars are wider than the other three — hence the code's name. The complete code is shown in table 1, along with a few practical examples in figure 1. The narrow bars stand for '0's, the (three times) wider ones represent '1's.
Reading errors are avoided in three ways. In the first place, exactly two out of every five bars must be wider. If not, the read-out comes to a halt. The second check concerns the number of bars: between the start and stop signs the five bar pattern appears a certain number of times, and so the total number of bars must be a multiple of five. The start and stop signs enable the computer to determine in which direction to read the data. If the optical 'pen' (or wand) with which the code is scanned moves across the code from right to left, the computer has to 'look' at the ones and noughts in the reverse order before being able to decode them.
The third check for mistakes involves a simple addition that may vary from one system to another. Usually, what it comes down to is that all the digits of the figure are added, after which the last digit of the sum is placed behind the figure (in other words, in front of the stop sign). This is shown in figure 1 as: 2+2+1+4=9. If the first two zeros had been an eight and a two, respectively, this would have given the same result: 8+2+2+2+1+4=19, as only the 9 is

Table 1

Character	2 out of 5 Bar Coda
0	00110
1	10001
2	01001
3	11000
4	00101
5	10100
6	01100
7	00011
8	10010
9	01010
Start	110
Stop	010

0 = narrow bar
1 = wide bar

1

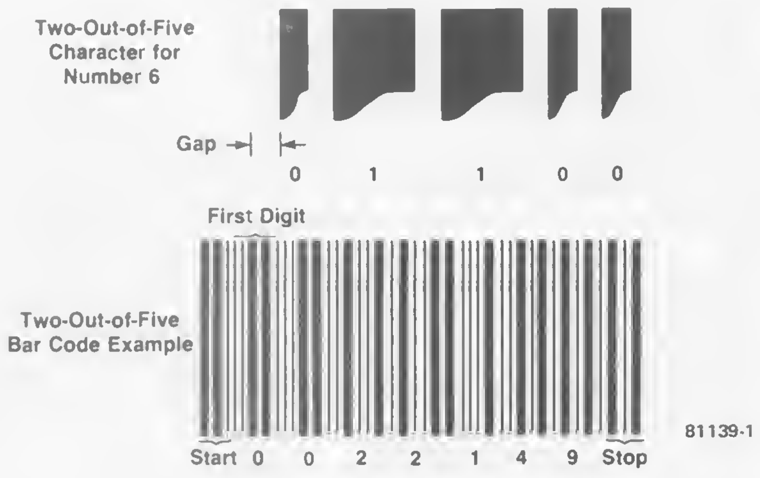
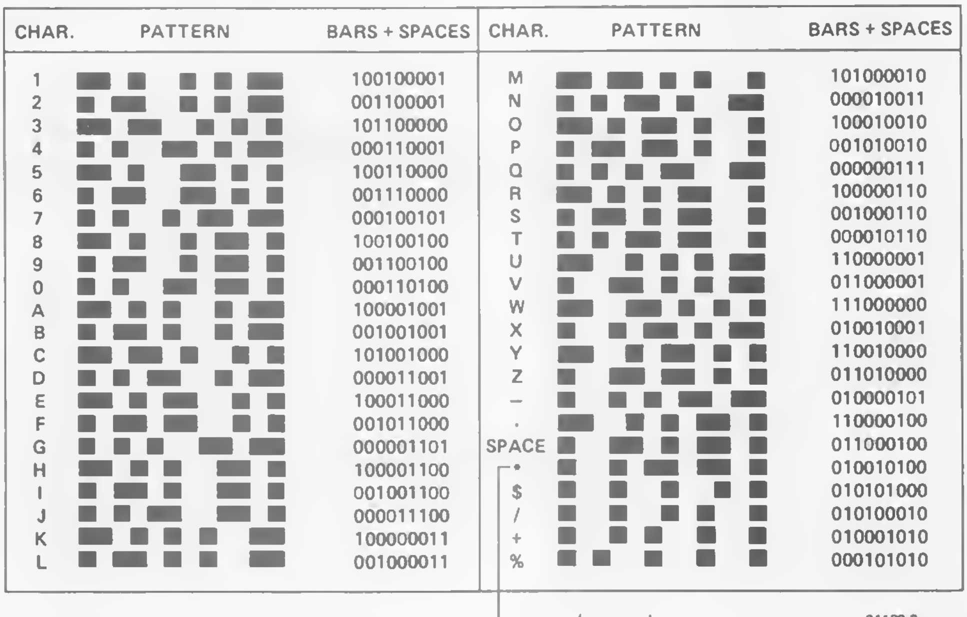

Two-Out-of-Five Character for Number 6
Gap
0 1 1 0 0
First Digit
Two-Out-of-Five Bar Code Example
Start 0 0 2 2 1 4 9 Stop
81139-1

Figure 1. In tha 'two out of fiva' coda one digit is represented by fiva bars, two of which are widar than tha othars.

added.
A much more complicated system is *code 39*. Here there are nine symbols (bars or spaces) per character, three of which are wider. In other words, this is a '3 out of 9' code which, with typical American brevity, is referred to as '39'. As can be seen in figure 2, the code provides many more possibilities: not only figures are represented, but also all the letters in the alphabet and several punctuation marks. A '1' is expressed as either a bar or a space that has twice the width of a '0'. The space between two characters has a fixed width of about one and a half times the size of a narrow space.
The two codes we have referred to both use five bars per digit (or letter). Jam jars, lemonade bottles and tins, etc. are usually labelled with a code that requires fewer bars than this. The *Universal Product Code* (UPC), for instance, only uses two bars per digit. This means the simple 'wide and narrow' system is no longer valid and both bars and spaces can now have up to four different widths.
The total width (bars plus spaces) for one digit is fixed at seven 'units'. This can be for instance a double bar, then a single space, then a triple bar and a single space=2+1+3+1=7 units. Two examples of this are shown in figure 3. Table 2 provides a survey of the code. The bars can be read in two directions, thus both 3-2-1-1 and 1-1-2-3 will be acknowledged as '0'. This fact is used in certain error detection systems.

2

CODE 39 Code Configuration



CHAR.	PATTERN	BARS + SPACES
1		100100001
2		001100001
3		101100000
4		000110001
5		100110000
6		001110000
7		000100101
8		100100100
9		001100100
0		000110100
A		100001001
B		001001001
C		101001000
D		000011001
E		100011000
F		001011000
G		000001101
H		100001100
I		001001100
J		000011100
K		100000011
L		001000011
M		101000010
N		000010011
O		100010010
P		001010010
Q		000000111
R		100000110
S		001000110
T		000010110
U		110000001
V		011000001
W		111000000
X		010010001
Y		110010000
Z		011010000
–		010000101
.		110000100
SPACE		011000100
*		010010100
$		010101000
/		010100010
+		010001010
%		000101010

* start/stop code

81139-2

Figure 2. Coda 39 features letters and a number of other signs as well as digits. In this particular case the spacas ara part of tha code (continuous). Threa out of nine bars and spaces ara three times wider.

Zebra crossing software

When a large quantity of (micro) computer software is to be made available to the public, the question that always arises is: how to distribute it. One way is to put it on paper in the form of a table using noughts and ones, or hexadecimal figures. The problem is that such printing matter is particularly prone to error – even when the original is produced as a computer print-out, it still must be keyed in by hand. Besides, it involves an awful lot of work and the final result is by no means aesthetically satisfying.

For some time now Elektor has been making use of an alternative solution, the 'software service', which consists of programmed records, cassettes and EPROMs. These are easy to use and lead to few problems. The only thing is, they're expensive.

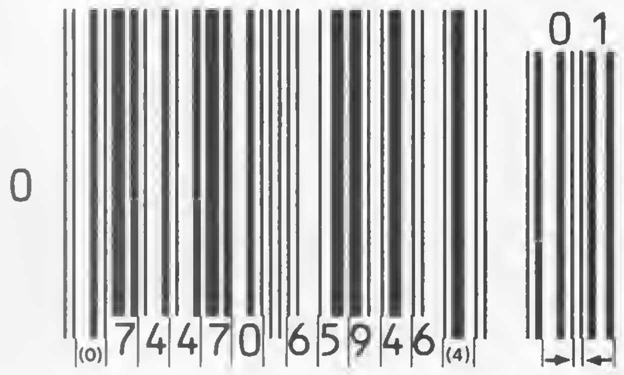
What could be better, theoretically, than a suitably adapted bar code system? A microprocessor could be 'taught' to print the bars (– a nightmare for the human draughtsman!). And, provided the user has a suitable read-out system at his disposal, the program could be entered into the computer in a few minutes.

This is exactly what a number of software suppliers are doing. Recently, Hewlett Packard started to provide its HP-41C calculator with bar code programs. Although the system HP uses is of course tailored to their application, it is a step in the right direction as far as the software service is concerned. Let's take a closer look.

As figure 4 shows, HP has selected the simplest code. The information is stored in the bars only: '0' is in the form of a narrow bar and '1' is a wide one. A single bar code line can represent up to 16 bytes, which are preceded by a start sign (two narrow bars) and end in a stop sign. The start and stop signs serve a double purpose: they enable the 'unit width' to be determined and at the same time they indicate the scan direction so that the recording unit knows whether or not to reverse the order of the entered bits. Elsewhere in this issue the entire read operation is considered in detail. At this stage we are only interested in the basic principle of the system.

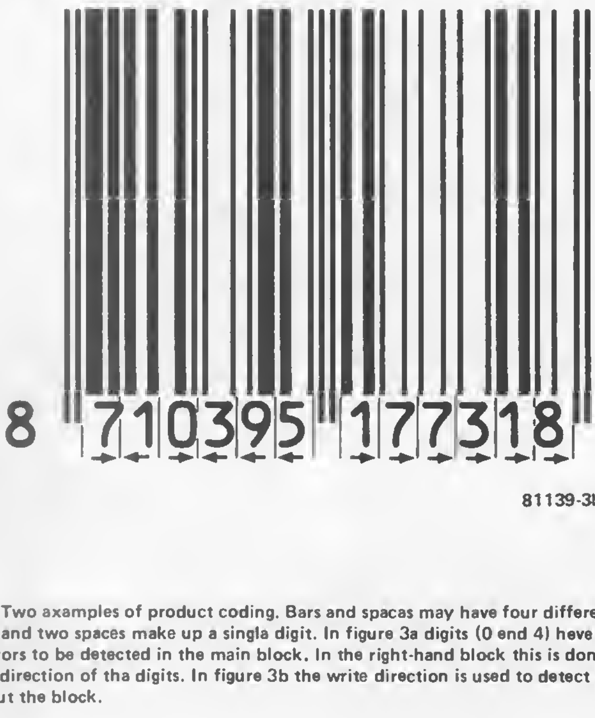
The whole idea behind the bar code system is to be able to read in data quickly by scanning the bars with a light pen. As this is done by hand it is unlikely to occur at, say, a speed of 7.5 cm per second ± 0.1%. A good system must be independent of the read speed over a large range and the speed may not be presumed to be anything like constant. This aspect is solved by using a new 'unit size' for decoding each bar. The unit size is derived from the immediately preceding bar and space. Reliable results can be obtained in this way for any reading speed between 76 cm per second and 7.6 cm per second.

3a



81139-3a

3b



81139-3b

Figure 3. Two axamples of product coding. Bars and spacas may have four different widths. Two bars and two spaces make up a singla digit. In figure 3a digits (0 end 4) heve been added to enable errors to be detected in the main block. In the right-hand block this is dona by chenging the write direction of tha digits. In figure 3b the write direction is used to detect errors throughout the block.

Tabla 2

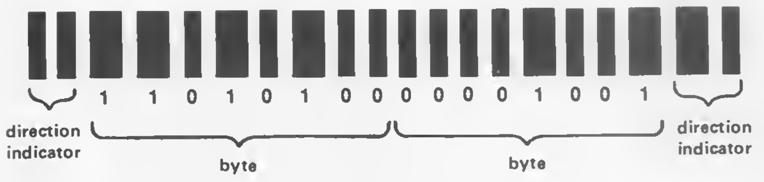
Charectar	UPC Bar Code
0	3-2-1-1
1	2-2-2-1
2	2-1-2-2
3	1-4-1-1
4	1-1-3-2
5	1-2-3-1
6	1-1-1-4
7	1-3-1-2
8	1-2-1-3
9	3-1-1-2
start/stop	1-1-1

The figures indicate the number of width units for the two bars and two spaces used in each character.

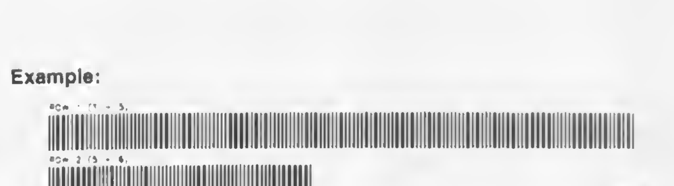
As to the scan direction, this is no problem at all. The entire sixteen bytes (128 bits) are stored in an intermediary memory during the read-out. The location of each bit in the memory is determined by a 'pointer'. This is reset at the beginning of a line. When reading takes place from left to right, the pointer too will run through memory 'from left to right'. If the scanner is moved from right to left, the pointer will travel in the opposite direction as well. Once 16 bytes have been read in, the pointer will only be reset if they were entered from left to right. Next, the bits are all read out of memory again one by one, with the pointer moving from left to right. This all seems rather complicated, so let's take a specific example.

If eight digits are written down from left to right, they can be read into an eight-digit intermediary memory according to the system described above. First from left to right:
1 2 3 4 5 6 7 8
In our mind's eye we can see the pointer being reset to the first digit and then it will be clear that reading from left to right will give the figures in the correct order.
If, however, the digits are entered from right to left things become a little more difficult. The first digit (8) will be placed at the far left, leaving no room for the others to precede it. What happens is that the pointer shifts to the left, outside the available range, and re-appears at the right-hand side! Thus, the next digit (the 7) will be placed at the far right, left of it the 6, then the 5, etc. The result looks like this:
8 1 2 3 4 5 6 7
The last digit to be entered was 1, and the pointer will remain in that position and will not be reset! The digits are then read from left to right and they appear at the output in the correct order.

4



81139-4a



Example:

81139-4b

Figure 4. The HP system operates according to two bar widths: narrow ones for '0's and wide ones for '1' s

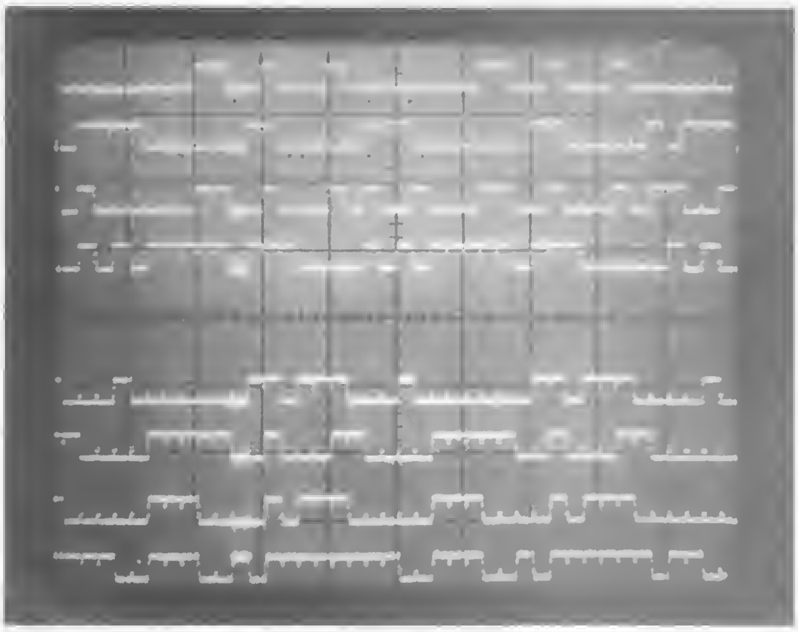
Pages and pages of bars . . .

Even if programs can be translated into bars, the question still remains: how much space would that occupy?
The HP system relies on using a reasonable light pen at a reasonable price. Its measuring error margin is less than 0.1 mm, so it can cope with even very narrow bars. For practical reasons, however, (print quality, wide read speed range) bars will have to have a minimum unit width or about 0.5 mm. Thus each bit (bar plus space) would occupy between 1 and 1.5 mm. Assuming each bit would on average take up about 1.25 mm, one byte (8 bits) would correspond to 10 mm. A 16 byte line including the start, stop and check data would therefore extend across one magazine page and the 32 lines required for ½ Kbyte of software would cover a whole page from top to bottom. In other words, the system is far from compact and is only viable for relatively short programs.
If Elektor were to use the system to publish its software, another solution might be to print about 80 bytes in the margin of each page. Thus, the entire chess computer program (4 Kbyte) would spread over 50 pages, so it would just about fit into one issue. Whether it would look good is of course another matter! ⧓

The previous articles covered the theoretical aspects of the logic analyser in reasonable detail. This article is intended to make the transition from theory to reality as simple as possible. It must be emphasised that care should be taken at every stage to ensure a reliable piece of test equipment.
To simplify matters the circuit is split into five printed circuit boards, consisting of a main board, three modules and a display. (A sixth board is meant for the power supply.) Each one is compact enough for the entire circuit to be incorporated in a reasonable size case. The main board also has room for an additional module which can be mounted at a later date and which will contain the storage scope extension.

logic analyser III

**With any project as complex as the logic analyser, the performance of the completed instrument depends to a very large extent on how it was put together. Poor constructional methods make even an exceptional design totally useless.
This, the final article on the logic analyser, puts the theory into practice. Full constructional details together with testing and calibration methods are included. A chapter devoted to the use of the logic analyser brings this part of the project to a satisfying conclusion.**



Constructing the printed circuit boards

Naturally, a design like this should only be built with good quality components, as it is a precision instrument. For the same reason it is important to fit the components with care and 'clean' solder joints.
Figures 1 . . . 5 show the main board, the input board, the memory board, the cursor board and the display board. It is best to mount the components in the following order: wire links, resistors, capacitors and semiconductors. IC sockets are strongly recommended, but make sure they're good quality types, otherwise they could make matters worse. A separate parts list is given for each board and this also indicates how many wire links are required. On some boards there are quite a few and it is very easy to leave one out by mistake. It should be noted that the main board only has room for small crystals; furthermore, the two dotted wire links should not be omitted.
The modules may be connected to the main board by means of flying wires, but of course it is neater to use connectors (DIN 41617 standard).
Before we go any further, a few things need to be said about the circuit diagrams which were published in last month's issue. A few minor drawing errors cropped up. Pin 7 of IC6 is grounded and not connected to +5 V. For MMV1, connection A (pin 9) is grounded and connection B (pin 10) is linked to CP2 (output of N23). In the IC power supply connections in the cursor diagram IC20 should read IC29 and pin 5 of IC31 should be connected to +5 V; pin 6 of this IC is not connected. Note that these corrections only

apply to the circuits given last month, the boards shown in this article are correct.

Back to building the thing... It is time to position the modules. The input board is placed closest to the components on the main borad, the memory is placed next to it and finally the cursor board. The copper track pattern side of the three modules should face the main board components. The remaining connections (or connector) can be used at a later date to mount the storage scope board.

So far so good. Now the power supply can be mounted (see figure 6). When constructing the supply, make sure that the supply-common connection of IC2 makes good contact with the board. A bad contact here can lead to 50 Hz ripple on the supply lines, which will appear as 'jitter' on the upper lines on the oscilloscope. The entire unit, including the mains transformer, can now be mounted in a suitable case. The cursor board can be left out for the moment, as this is not needed at this stage. A front panel design is shown in figure 7. Covering the displays with a piece of red perspex gives a clear read-out.

Now the wire links may be made between the boards and the switches. These should be as short as possible. Potentiometer P1 is connected to the main board with a length of screened cable. The screen must be connected to the +5 V supply.

The measuring cable for the data, clock and trigger inputs consists of a piece of 23 way ribbon cable, not more than 40 cm long. A ground lead is situated between all the input leads; these ground leads are interconnected at both ends of the cable. How this should be done is shown in figure 8.

The front panel layout shown in figure 7 includes switches to operate both the logic analyser and the storage scope. The controls for the latter are located to the left of the test cable output.

For the time being (until the storage scope extension is added) the dotted wire links (Y-Z and E-⊥) must be included.

Testing and calibrating the circuit

The power supply is not connected to the main board for the time being. To start with, the mains is switched on and the power supply is tested to see whether it is in fact producing 5 V (if the relevant components are fitted, the +12 V and –5 voltages may be checked at the same time). Then the power supply is switched off and connected to the main board. When it is switched on again the clock generator section may be checked. Now the oscillator may be calibrated, if C7 and C8 are mounted instead of a crystal. If the two capacitors have been replaced by a 4 MHz crystal, no calibration is required. If you're fortunate enough to own a frequency meter, the circuit is very easy to adjust. Connect the meter to the centre contact S1 and place S1 in position A. Take care that the circuit is not triggered in any way; in other words, the LED must remain unlit (depress S16). Now trimmer C8 is adjusted so that the frequency meter indicates exactly 4 MHz.

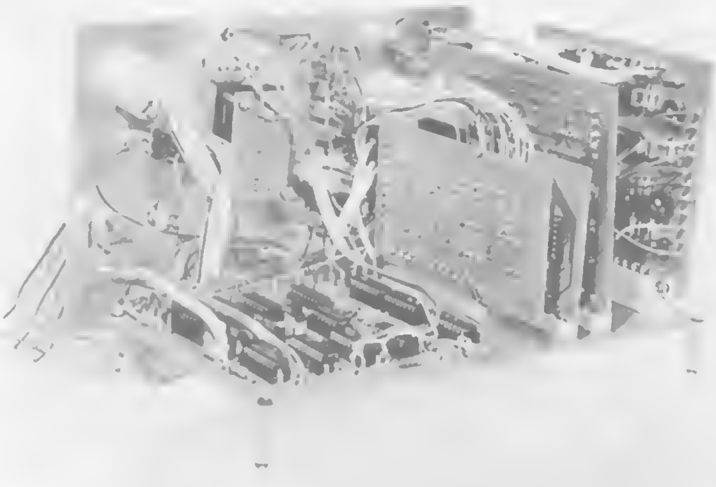
Don't worry if you haven't got a frequency meter, there is an equally good alternative. Switch S1 to position J and S2 to b. The centre contact S1 is connected to the oscilloscope — this you must have, otherwise there's no point in building the logic analyser at all! — and C8 is adjusted so that the frequency at that point is 200 Hz with C8. After this the Y input of the oscilloscope is connected to one of the transformer inputs on the power supply board and the external trigger input of the scope is linked to the centre contact of S1. The oscilloscope is switched to 'external triggering'. Now C8 is readjusted slightly until a stationary picture is obtained. In this way the clock generator can be adjusted to the same accuracy as the mains (better than 0.5%).

Let us now connect the oscilloscope to the outputs of the logic analyser and see what will be written onto the screen. Connect the Y and trigger inputs of the oscilloscope, set its sensitivity to 0.2 V per division, the input amplifier to DC

Technical data:

* Clock:
  - internal, presettable between 250 ns and 5 ms;
  - external, maximum 4 MHz (presettable on either positive or negative edge).
  - Clock delay is presettable tebween 100 and 600 ns.
* Memory: 256 bytes
* Inputs:
  - 8 data lines (1 LS TTL load);
  - 2 external trigger inputs (2 LS TTL load max.);
  - external clock input (1 LS TTL load)
* Triggering:
  - via data and trigger lines, each line can be individually set to LOW, HIGH or DON'T CARE
  - manual trigger
* Trigger alternatives:
  - post trigger
  - centre trigger
  - pre trigger
* Cursor with hexadecimal display

Type of oscilloscope required:
  - single channel;
  - external trigger input;
  - bandwidth > 500 kHz,
  - Z modulation input, (optional).



Parts list main board

Resistors:
R1,R2,R12,R15,R29 = 5k6
R13 = 470 Ω
R14,R21 = 2k2
R16 = 22 k
R17 = 3k3
R18 = 56 k
R19 = 18 k
E20,R30 = 4k7
R22,R24,R28 = 820 Ω
R23 = 390 Ω
R25 = 1 k
R26 = 6k8
R27 = 10 k
R36 = 33 k
P1 = 50 k

Capacitors:
C2 = 4p7
C3 = 47 n ceramic
C4 = 120 p
C5 = 22 p
C6 = 47 p
C7 = 3p3
C8 = 1 . . . 6 p trimmer
C16 = 100 n
C17 . . . C20,C22 = 22 n
C26 = 100 μ/10 V

Semiconductors:
T1 = TUN
D1 = LED
IC6 = 74LS151
IC7,IC8 = 74LS163
IC9 = 74LS324
IC10 = 74LS123
IC11 = 74LS109
IC12,IC13,IC14 = 74LS390
IC17 = 74LS266
IC18,IC20 = 74LS132
IC19 = 74LS32
IC28 = 74LS73

Number of wire links: 24

Miscellaneous:
S1 = single pole, 12 position switch
S2 = three pole switch
S3,S19 = toggle switch
S4 = double pole three position switch
S5 . . . S14 = three position switch
S15,S16 = pushbutton switch
S17,S18 = pushbutton switch

1

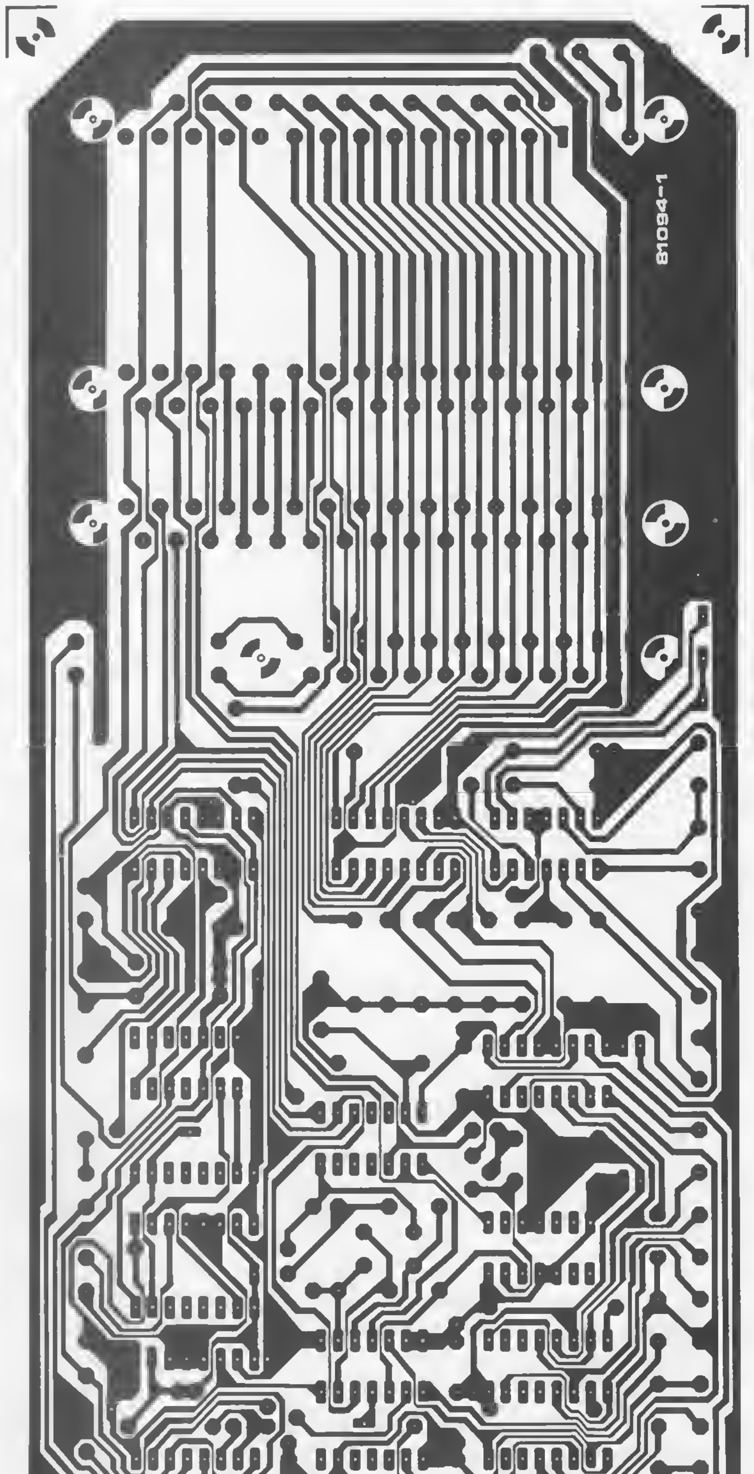
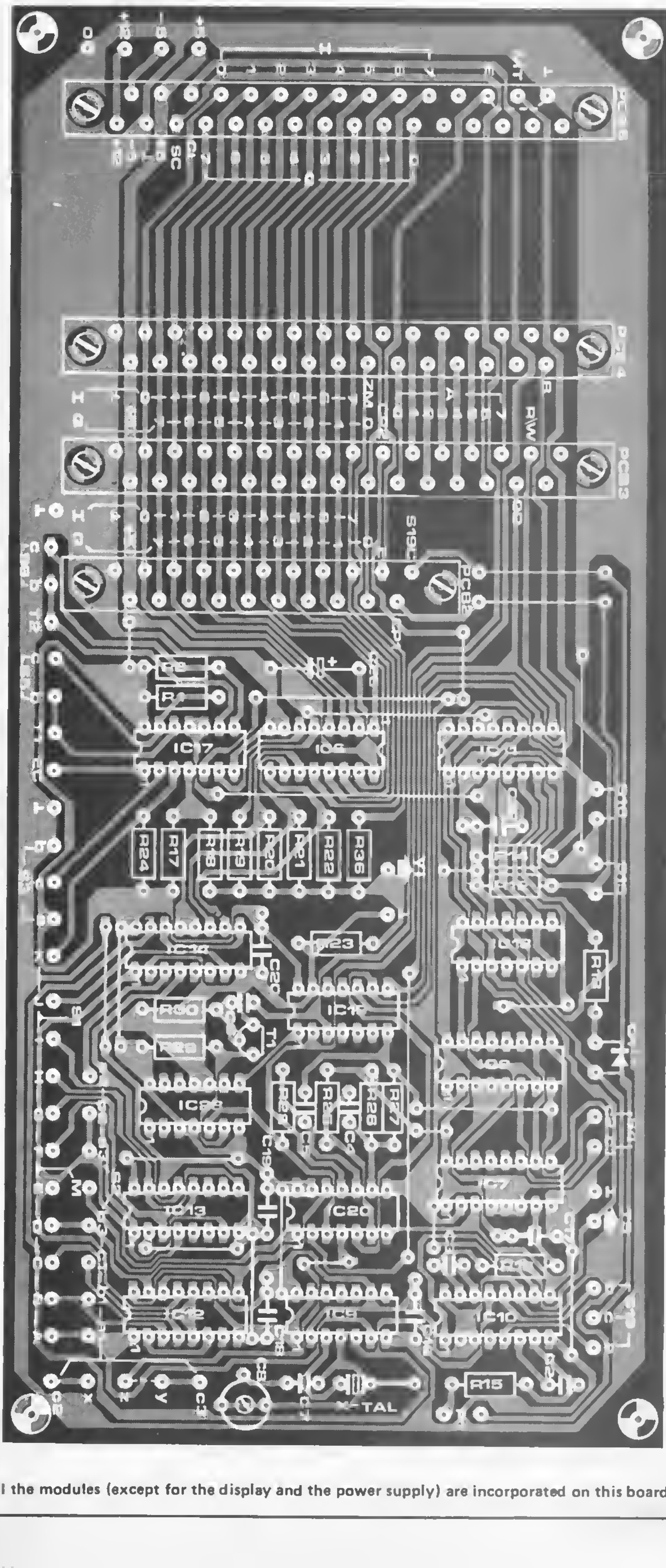


Figure 1. The copper track pattern and component layout of the logic analyser's main board.



All the modules (except for the display and the power supply) are incorporated on this board.

and the time base to 0.1 milliseconds per division. Switching S1 of the analyser into any position (except K) and depressing the 'manual trigger' button, should cause eight lines of random data to be written on the screen. The time base of the 'scope is then set so that all 256 bits appear on the screen; in other words, the time it takes a full line of data to be written across the screen must be about 128 ms.

Once this has been done the cursor board may be added. The power supply is switched off and the board is mounted on the main board. A length of ribbon cable is used to connect the display to the main board. An alternative is to use two small 16 pole DIL connectors. Switching the analyser on again and triggering it manually should again cause random data to appear on the screen. If R36 was duly mounted on the main board each line will be seen to be 'dented'. Operating the up and down keys changes the readout and moves the dents left or right.

By the way, the cursor and display board can be omitted without having to modify anything. This reduces the total cost of the project, but on the other hand, the cursor can be a very useful aid.

That completes the logic analyser, except for a few minor details. First the clock delay time. When S19 is in position 'a' this will be about 100 ns and in position 'b' it can be adjusted between 200 and 600 ns by means of P1. The potentiometer can be provided with a suitable scale. Readers who own a two-channel oscilloscope with a bandwidth of at least 20 MHz can measure the delay between the positive edge of the signal at the centre contact of S1 and the positive edge of the signal at contact 'b' of switch S19 (assuming that S1 is in position 'c' and S2 is on 'b'). The times may then be indicated next to the various positions of P1.

If the oscilloscope is found to be unsuitable for this procedure, P1 may be calibrated with the aid of the circuit in figure 9. This consists of a D type flipflop of which the D input is connected to S19b and the clock input to the centre contact of S1. A multimeter, set to 5 . . . 10 V DC range, is connected to the Q output. S1 is now switched to position 'A' and P1 is set at minimum resistance. The meter should now indicate a voltage between 2.7 and 5 V (the Q output is logic 1). P1 is turned up slowly until the meter just drops back to zero, where the delay will be 250 ns. Next, S1 is switched to 'B' and the same procedure is repeated. The delay at P1's present position will be 500 ns. The area between the two values may be split up into equal divisions since the potentiometer operates in a fairly linear manner.

A further point only concerns readers who have an oscilloscope without a variable time base. In other words, the final 56 bits of each line do not appear

2

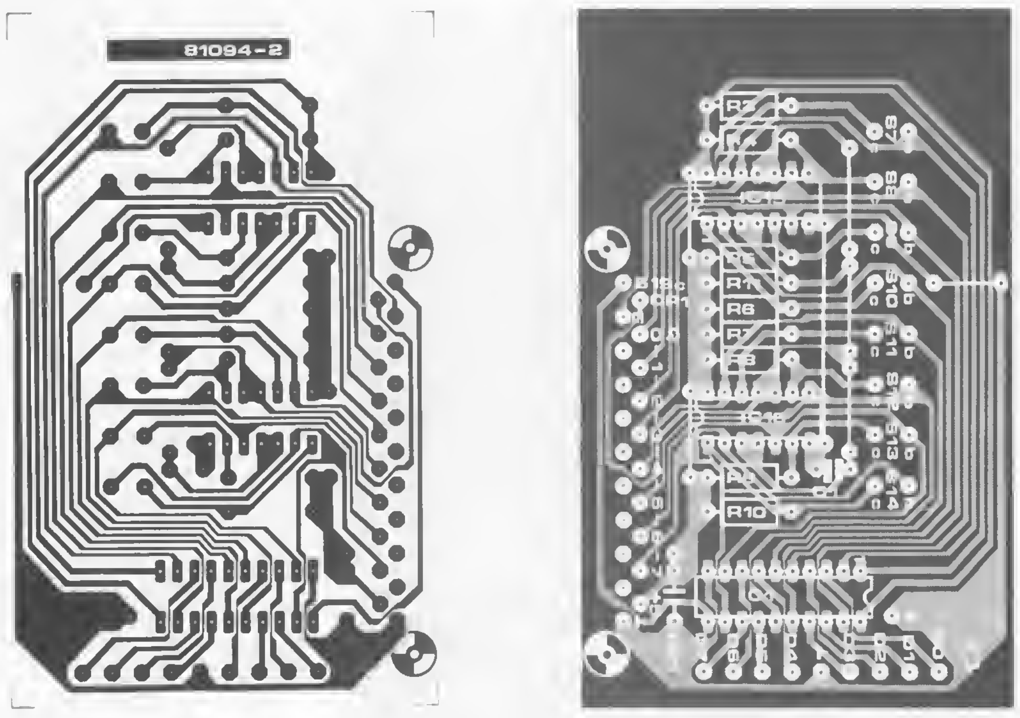


Figure 2. The input board to which the data lines and the word recognizer switches S7 . . . S14 ere connected.

3

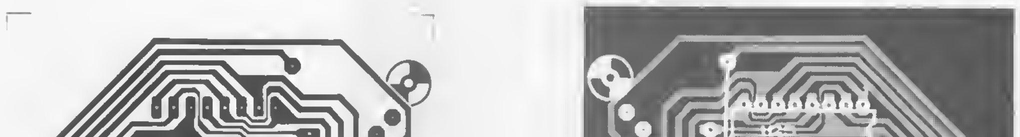


Figure 3. The memory board includes two RAMs and two counters which together constitute counter A.

Parts list input board

Resistors:
R3 . . . R10 = 5k6
R11 = 820 Ω

Capacitors:
C1 = 39 p
C21 = 22 n

Semiconductors:
IC1 = 74LS374
IC15,IC16 = 74LS266

Number of wire links: 7

Parts list memory board

Capacitors:
C14,C15,C25 = 22 n

Semiconductors:
IC2,IC3 = SYP 2101A-2
(Ta = 250 ns)
IC4,IC5 = 74LS191

Number of wire links: 6

Parts list cursor board

Resistors:
R31 = 1 k
R32 = 2k7
R33 = 4k7
R34,R35,R40 = 820 Ω
E37,R38 = 680 Ω
R39 = 470 Ω
R41 = 10 k
R42 = 22 k

Capacitors:
C9 = 180 p
C10,C11 = 47 μ/10 V
C12 = 27 p
C13 = 1 n
C23,C24 = 22 n
C27 = 390 p
C28 = 10 μ/10 V

Semiconductors:
T2 = BC 517
IC21,IC22 = 74LS266
IC23,IC24 = 9368
IC25,IC26 = 74LS191
IC27 = 74LS122
IC29 = 74LS132
IC30 = 74LS73
IC31 = 74LS32

Number of wire links: 9

Parts list display board

Miscellaneous:
LD1,LD2 = HP 5082-7760
(DL 7760)

4

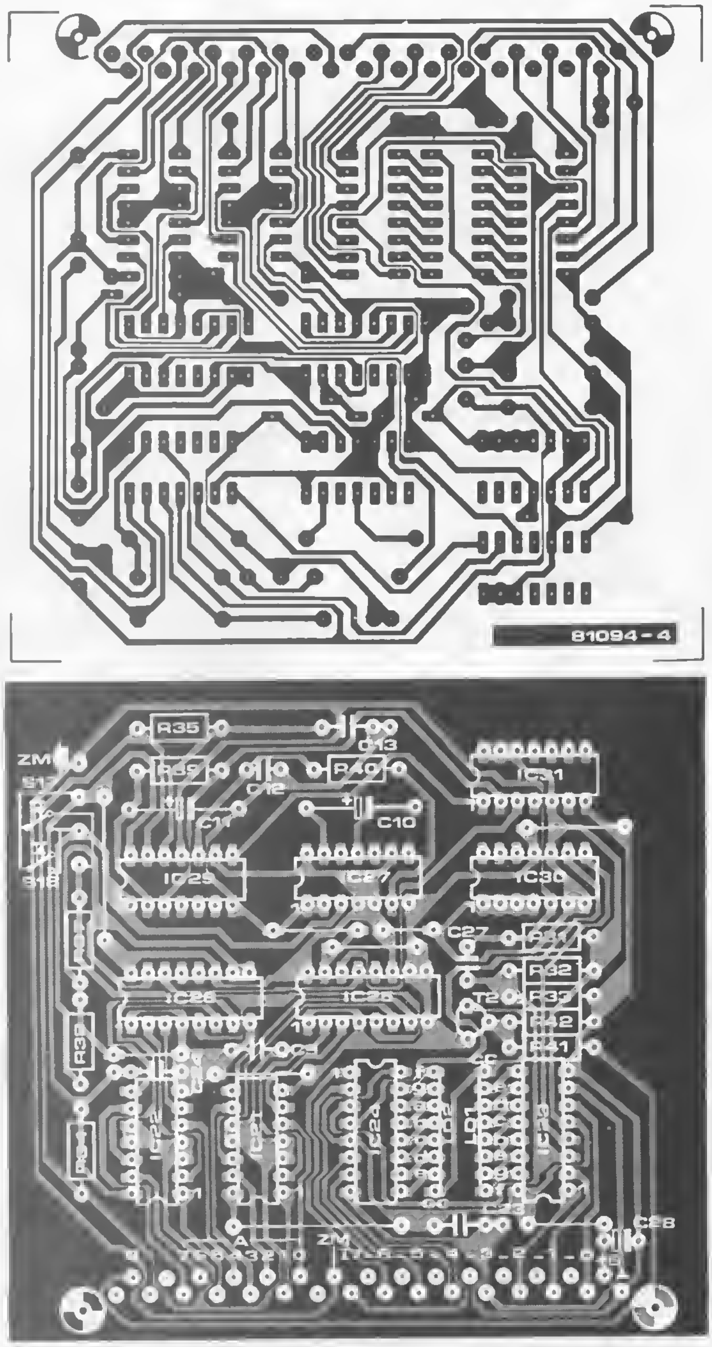


Figure 4. The copper track pattern and the component overlay of the cursur circuit.

5

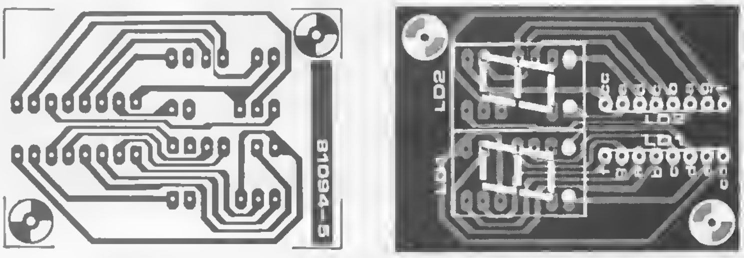
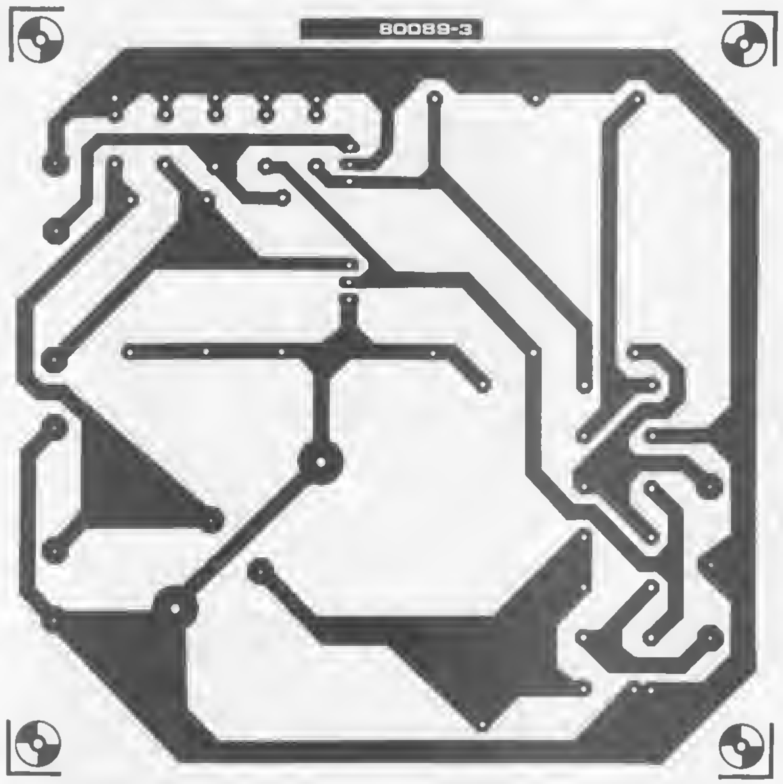


Figure 5. The display board.

6



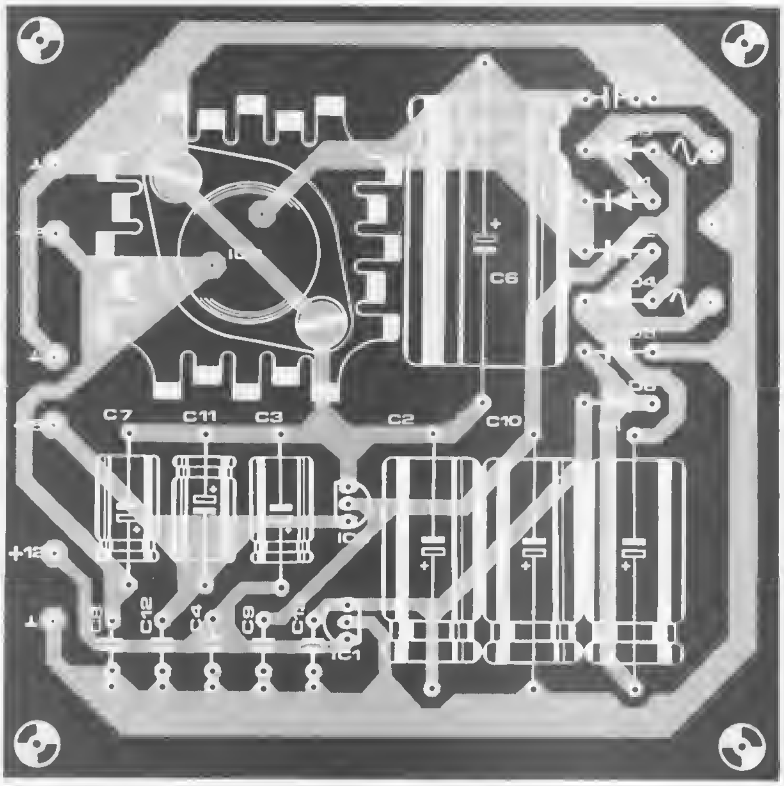


Figure 6. The power supply board was designed for the Junior Computer and is ideal for the logic analyser. Only the 5 V section is required until the storage scope extension is built.

Parts list power supply board

Capacitors:
C5 = 2200 µ/25 V
C6,C8 = 100 n
C7 = 100 µ/25 V

Semiconductors:
D1,D2 = 1N4004
IC2 = LM 309

Miscellaneous:
Tr1 = transformer 2 x 9/10 V/1,5 A
S1 = double pole mains switch
F1 = 500 mA fuse, with fuse holder
Heat sink for IC2
For the storage scope-extension!

Capacitors:
C1,C2,C9 = 470 µ/25 V
C3,C11 = 47 µ/25 V
C4,C10,C12,C13 = 100 n

Semiconductors:
D3,D4,D5,D6 = 1N4004
IC1 = 78L12
IC3 = 79L05

on the screen. Moving the picture in the x direction has no effect, as the moment of the trigger pulse is predetermined. Using the circuit in figure 10, the trigger pulse may be delayed and the final 56 bits can be brought into view by shifting the picture in the x direction. This circuit can be included between the trigger output of the analyser and the trigger input of the oscilloscope. The lines can be moved across the screen by turning pot P1 and so any section of information may be made visible.

How to use the logic analyser

It is of course essential to know what all the controls on a measuring instrument are for and how to operate them. Before we deal with an actual example, let us review the various switches. S1 and S2 select the sampling time. As you will remember, the previous issue contained a table giving these times. With S1 in position 'K' an external clock may be connected and S3 determines whether the analyser is to sample on the positive or the negative edge of the external clock signal.

Switch S19 (delay) and pot P1 (clock delay), set the delay between the clock signal and the actual data entry in the latch. This delay is only relevant if an external clock signal is being used. When

using the internal clock, the delay switch S19 must always be in position 'a' and P1 is turned fully anti-clockwise.
A word should be said about the external clock input. At 2 MHz frequencies and higher the input clock signal must be symmetrical if the RAMs are to work properly. At frequencies above 1 MHz (both internal and external clock signals) the delay time should not be too long, as this would lead to an incorrect data reading. In general, the delay time must always be less than the clock pulse length (that is, less than half the clock period).
The word recognizer switches have already been dealt with in great detail: they preset the required trigger word. The switches for the two external trigger inputs work in exactly the same way: logic one, logic zero or don't care.
S15 is used for manual triggering. The moment that the circuit receives a trigger pulse the LED lights.
The position of switch S4 (trigger mode) determines when data entry stops. In the 'post trigger' state the 255 bits following the trigger pulse are stored in memory; 'centre trigger' means that 125 samples preceding and 129 samples following the trigger pulse are stored; finally, in the 'pre trigger' mode the 255 preceding samples are stored.
When the reset button is operated, the analyser starts to read in new data and again reacts to the preset trigger 'word'.
The two pushbuttons that operate the cursor are straightforward enough. These move the cursor left and right across the screen.

Now for a practical example. Let's assume that we wish to examine the data bus of a microprocessor. We would then connect the eight inputs of the analyser to the corresponding data lines. The external clock input can be linked to the clock of the microprocessor and the required trigger word is preset with the word recognizer switches. The trigger mode switch is set according to the desired data.
After depressing the reset button the microprocessor is started. When the programmed trigger word appears on the data bus, the circuit will trigger and the stored data sequence will appear on the screen of the oscilloscope. The data bytes can be scanned in hexadecimal form on the display.
Let's put this into practice using the Junior Computer's data bus. The external clock input of the analyser is linked to Φ2 and S3 is set to position 'a' (positive edge). S1 must now of course be in positoin 'K', S19 in position 'b' and P1 sets the delay to about 400 ns. Other processors naturally require different presets and it is not possible to give any fixed rules, as various factors, such as the clock frequency and the RAM and ROM access times, are involved.
This completes the description of the logic analyser. Look out for the storage scope extension in a forthcoming issue!

7

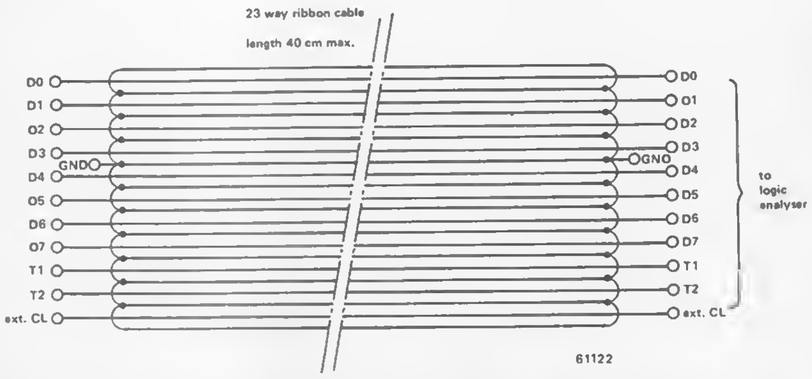


Figure 7. This shows the position of the ground leads in the test cable. These leads ere interconnected et both ends of the cable.

8

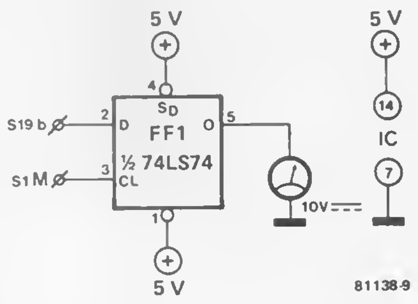


Figure 8. This euxiliary circuit can be used for calibrating P1.

9

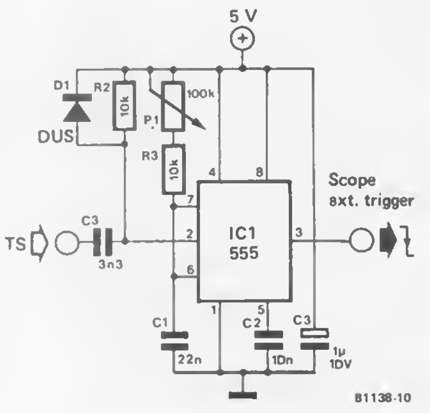


Figure 9. If readers happen to have en oscilloscope without a variable time bese, they can bring all the bytes onto the screen by edding this circuit to their enelyser.

waveform generator

Square wave, sine wave, saw tooth, triangle, symmetrical, asymmetrical . . . you name it, this generator can produce them all. It is hardly surprising that such devices are so popular, for they are indispensable in the lab. This time Elektor has come up with one that can be used both in analogue and in digital technology.

The main feature of this particular circuit is that it can be built with non-critical parts at low cost and yet guarantee a highly satisfactory performance. In addition, it is a vast improvement on its conventional counterparts. These can suffer from many disadvantages. The commonly used Wien oscillator, for instance, requires a stereo potentiometer to adjust the frequency. The tolerance in this component inevitably leads to distortion and a further disadvantage is the fact that the amplitude tends to vary with a change in frequency.

In the circuit described in this article, advantage is taken of the fact that a symmetrical square wave possesses a great many unequal harmonics. It will be seen that an RC filter enables a very clean sine wave to be 'distilled' from such harmonics. Depending on the frequency, the generator's distortion is somewhere between 2% and 0.75% and the slew rate is about 10 V per second, which is not bad at all. Let us now consider the basic circuit diagram in figure 1. Opamp IC1 is part of a tuning network. This is modulated by a square wave obtained from comparator IC2. The frequency is determined by R1, R2, C1, C2 and R3. Pot R3 is used to trim the network, and so adjust the output frequency. Tuning the filter does not affect the gain or the band width. The amplitude will remain constant and so will the resulting sine wave.
The circuit can be made to oscillate by feeding the square wave back to the input of the filter. Zener diode D1 stabilises the square wave.

1

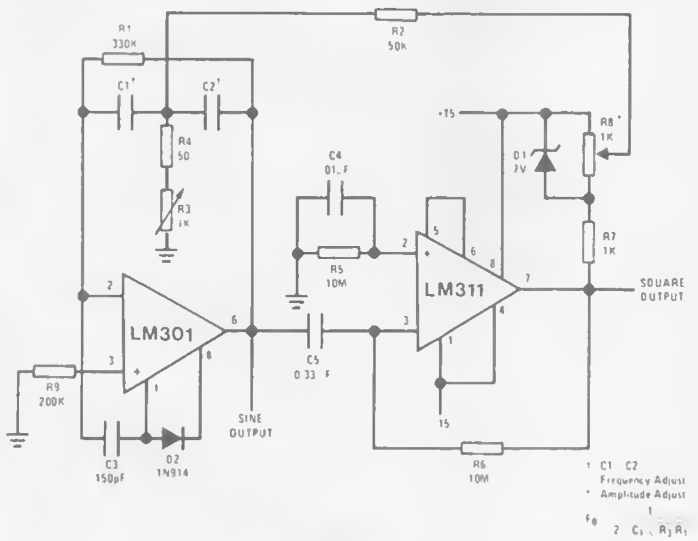


Figure 1. The main principle behind the square/sine wave generator. The scale ranges from 18 Hz to 50 kHz.

The circuit diagram

Figure 2 shows the complete circuit diagram, which is different from figure 1 in that it features a range that has been divided into small steps. This gives a greater resolution to the overall frequency range. The various ranges are indicated in figure 4.
A further addition, as compared with the basic circuit, is the voltage divider around P3. This produces a variable DC voltage level that can be switched to the output by way of S3. This is especially important where digital circuits are concerned, as the zero level of the output waveform can now be made positive so that only positive pulses are generated. S4 has been included to allow preamplifiers to be connected without input capacitors. In the AC position only a genuine AC may pass

2

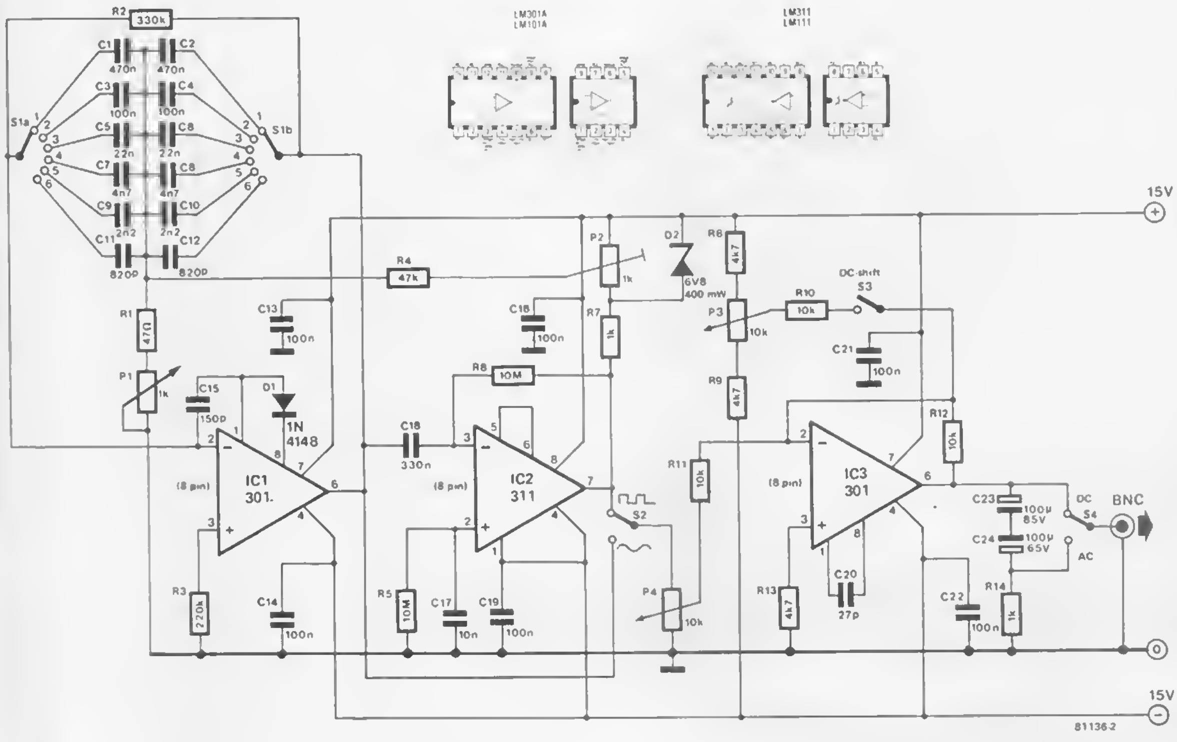


Figure 2. This is the complete circuit diegrem of the square/sine wave generator. The use of an oscillator or a frequency meter is essential to calibrate the 6 scales.

3

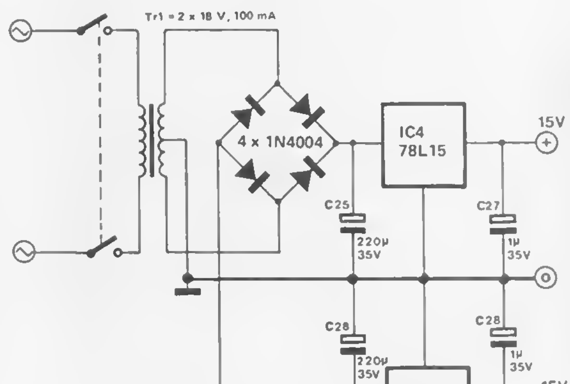


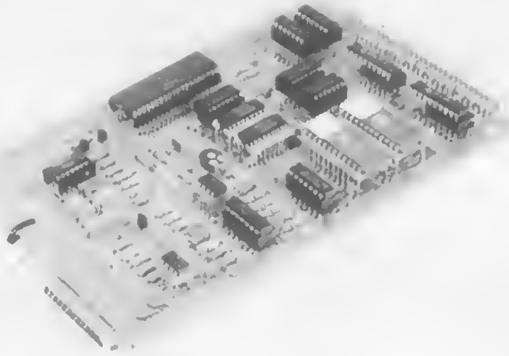
Figure 3. A suitable power supply for this particuler design.

4

S1a,b	1 = 18 Hz . . . 80 Hz
	2 = 80 Hz . . . 380 Hz
	3 = 380 Hz . . . 1,7 kHz
	4 = 1,7 kHz . . . 8 kHz
	5 = 4,4 kHz . . . 20 kHz
	6 = 20 kHz . . . 20 kHz

Figure 4. By dividing the total range into six scale divisions the scele resolution is increesed. This greetly simplifies the frequency adjustment.

the output; in the DC position the AC is superimposed on a DC.
Figure 3 shows a suitable symmetrical mains power supply. It is stabilised by two 78L15 ICs. The diagram includes a BNC connector at the output of the square/sine wave generator. As this is a low frequency generator, however, ordinary banana plug sockets may also be used. ◼



sophisticated software for the Junior Computer

two kilos of brain power, please!

Now that we've added to the Junior's hardware (see the relevant article elsewhere in this issue), we'll have to provide the accompanying software as well. For this Junior was sent to school where the machine assimilated two highly informative programs, 'TAPE MANAGEMENT' and 'PRINTER MONITOR'. The first of these enables data be read and written on tape and the second allows the Junior Computer to be connected to the Elekterminal or to a printer. Tape Management deals with function keys on the standard Junior Computer keyboard and Printer Monitor activates such keys on either the terminal or the printer keyboard. This article is nothing more than a short preview of the in-depth software description to be published in Book 3.

What is involved aren't exactly 'new' keys, but brand new functions that play a key role in the cassette interface. This means the 'old' keys must now bear yet another set of inscriptions. Now that the extension card has been added to the standard Junior Computer, programs can be entered within a much larger address range running from 0200 up to 07FF without interruption. That amounts to 1536 bytes which, thanks to the recording power of the cassette interface, only have to be entered once.

The cassette: a magnetic RAM with magnetism

The Junior Computer's horizons can be widened at the reasonable cost of a simple mono cassette recorder (from £10) and a couple of cassettes. If the C-60 type is used, about 25 minutes will be available per side, as allowance has to be made for breaks (3-4 minutes) and a short space at each end of the tape. This means that at a transmission speed of 50 bytes per second (we'll come back to that later) 25 x 60 x 40 = 73 kbytes (1 k = 1024 bytes) can be stored. That'll do for a start . . .

How is the data stored on tape?

Figure 1 looks rather like a train with a series of compartments in which different information is stored. The data does not necessarily belong to a complete program, it may also constitute a separate unit, a table or a piece of text, etc. In either case, however, a *data block* is involved.

Now let us look at the configuration in figure 1:

1. Considering the tape from left to right, the first 'block' contains 255 synchronisation characters. These filter out the actual beginning of a data block from other information that the Junior Computer would find hard to digest. In other words, don't try to make the machine 'eat' your baby's first babblings, your brother's trumpet voluntary or such remarks as: 'This is my first program on cassette' . . . the Junior Computer is not a parrot!!

N.B. Data is always in ASCII code, in 8 bit words when it is stored on tape. The furthest bit to the left is reserved for special functions and in this case will be zero. The bits of an ASCII byte are stored one behind the other, *serially*. The ASCII code of a synchronisation character is 16 in hexadecimal.

2. The initialisation character '*': its purpose is to signal when the series of synchronisation characters has gone by and the data itself has arrived (hexadecimal 2A in ASCII code).
3. The program number ID: this enables one program to be differentiated from another, thus ID really stands for ID here! Since 254 different program numbers are possible, that includes all the values between 01 and FE. Values 00 and FF fulfil a specific task when data is read from tape.
4. The low order address byte SAL:

1

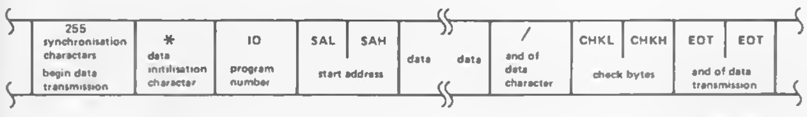


Figure 1. This is what a deta trensmission looks like on tape. It consists of a block pracaded by 255 synchronisation characters, the data initialisation cherecter, the program number end the start addrass, and followed by an and of dete cheracter, chack bytes and two end of deta transmission cheracters. This taping procedure is similer to that in the KIM computer, the only difference being thet the Junior featuras 255 synchronisation charactars, wharees the KIM has only 100.

corresponds to the first memory address belonging to the program or data block that is to be written on tape.
5. The high order address byte SAH.
6. The actual data block: each data byte is transcribed into two ASCII bytes, that is to say, one ASCII byte per nibble. The data byte is then stored on tape as a series of 16 bits. The write operation begins with the start address (SAH, SAL) and ends with the byte stored at address EAH, EAL minus one.
7. The end of data character "/" which indicates that all the data have been transmitted (ASCII: 2F).
8. The check bytes CHKL and CHKH: these make sure nothing was mislaid or misread during the recording. A cassette tape has a special gift for distorting information, inspite of the improved PLL (see the article on the hardware). It is therefore absolutely necessary to check whether one or more bits did not happen to be viciously mutilated in some invisible snare, the best method being to count them like the shepherd counts his sheep.

At the beginning of a write operation, the CHKH/CHKL bytes are zero. From SAL on, the bytes are all added to each other before being transcribed into the ASCII code. The program number (ID) is not taken into account here. Each time the contents of CHKL reach FF, that of CHKH is incremented once and when this too reaches FF, both are reset without causing any complications. When data is read on tape, the same procedure is used. All that has to be done then is to compare the result of the two counts: the sum of bytes during the write operation must equal that during the read operation, and vice versa. If so, there is reason to believe (and hope!) the transmission has been carried out correctly. If not, the tape will have to be tidied up here and there. We could of course provide you with a whole chapter of error statistics and other theoretical titbits . . . but at this stage it is better for readers to cross that bridge when they come to it.

With regard to the interface check system, let us compare it to a bank to illustrate its function. It is in the bank's (and our!) common interest to see that people's money is well looked after. It would therefore be disheartening, to say the least, for the employees to discover one morning that one of the clients had remained in the building after closing and had subsequently absconded with the cashbox. This can be avoided quite simply by counting all the clients who have come in and gone out during the day and then compare the two numbers. If they do not tally, then there is reason for worry. It could mean the counting system can't count, or that two persons passed the detector at exactly the same moment and were therefore registered as only one (what a primitive system, get a refund!), or even that a client cashed in on his Pool winnings and was shown out discretely by the manager through a door at the back. If two of the cases mentioned happened to coincide they would automatically cancel each other out and so would escape notice altogether.

As far as CHKL and CHKH are concerned, counting the bytes by adding them is, believe it or not, a perfectly foolproof method, inspite of all the peculiar circumstances that might arise.
9. Two EOT (End Of Transmission) characters which indicate that a data block has been transmitted (ASCII: Ø4).

How to write on tape

As we said before, data is transmitted bit by bit. In figure 2 pulse trains are shown with long wagons (relatively high frequency) and short wagons (relatively low frequency). A low level bit is made up of four half periods of 2400 Hz and a high level one consists of three half periods of 3600 Hz (figures 2 and 2a). A logic zero consists of six 3600 Hz half periods and two 2400 Hz half periods. The total length of the train remains unchanged, irrespective of whether the logic level is high or low. (This can be expressed as 9T where T equals the duration of a half period of 3600 Hz.) Furthermore, the train will always start at the higher frequency. The duration ratio for the high and low frequencies is either 2:1 or 1:2.

Figure 2b shows what happens in the KIM. The graph had to be spread across several lines, so please follow the arrow. Here a high logic level bit consists of nine full periods of 3700 Hz (which in this instance will be rounded off to 3600 Hz) plus twelve full periods of 2400 Hz. A low logic level bit, on the other hand, is made up of eighteen full 3600 Hz periods plus six 2400 Hz periods. From these figures it can be seen that a bit in the KIM lasts six times longer than one in the Junior Computer. The reading and writing speed is therefore much slower, but this has been remedied by the HYPERTAPE program written by Mr. J. Butterfield, which enables the transmission speed to be accelerated considerably, so that one KIM bit then equals one in the Junior Computer. The software in the Junior's DUMP/DUMPT writing routine differs from that of HYPERTAPE in a number of fundamental aspects. To look at this in detail would hardly suit the purpose of the present article, but not to worry, several pages are devoted to the subject in Book 3.

To get back to what we were saying, one bit of cassette software in the Junior Computer lasts 9 half periods of 3600 Hz, which is 9 x 139 = 1250 μs. In other words, 800 bits (or 100 ASCII characters = 50 data bytes) per second.

Data is written on tape during the DUMP/DUMPT routine. During this write operation the six displays remain unlit. The parameters for a successful operation are:

a. indicate the program number ID (Ø1 . . . FE; ØØ and FF serve a special function, which we'll come back to later)
b. indicate the start addresses SA
c. indicate an end address EA

How to read on tape

To return to figure 2a, this also contains signals which can be found at the PLL output. The PLL is activated when data is read on tape, as was mentioned previously in the hardware discussion, where we also referred to the 1 to 3 and the 2 to 4 transition. The PLL's output signal clearly shows how the ratios 2:1 and 1:2 may be obtained.

According to duration of the PLL output's high and low levels, the software will filter out either high or low level bits from the PLL signal. It is not the durations themselves that matter, but their relation to each other: is the 3600 Hz period (high PLL) any longer than the 2400 Hz (low PLL), or vice versa? If the 3600 Hz lasts longer than the 2400 Hz, the bit in question will be logic zero, otherwise it will be logic one. Since the durations themselves are immaterial, the Junior Computer may be used to read back periods taped on the KIM, inspite of the pulse length differences between the two systems. Consequently, signals 3 and 4 in figure 2a are identical to those in figure 2b, even though the signals in the second figure (2b) are six times longer than those in 2a. This is a great advantage, even for readers who do not have occasion to use the KIM, as it means that variations in the reading or recording speed (usually

4.75 cm per second) will not affect the transmission quality — there will be no 'flutter', etc. In other words it does not matter which type of cassette recorder is used and it is possible to tape on one recorder and play back the cassette on a different one altogether. This is because the 2:1 (low logic level) and 1:2 (high logic level) ratios are too wide apart to cause confusion.
N.B. PLL jitter is not taken into account during a data reading (see the hardware article for more details).
Data is read with the aid of the RDTAPE routine, which is called during the TAPE MANAGEMENT program. The two right hand displays of the Junior Computer conveniently indicate what is going on during the reading. The remaining four will stay unlit.

The first drawing in figure 3 applies when:

a. the tape passing the reading head (cassette is on 'play') contains no digestible data (space between two data blocks, empty tape, Beethoven's 5th, etc.). D5 and D6 will be seen to flicker during this period.
b. the tape contains a data block which is being read, but the beginning is missing, or the ID number does not correspond to the one specified. D5 and D6 will no longer flash but will remain constant. This situation is in force when the computer is in the synchronisation phase. In other words, it is reading the synchronisation characters preceding a data block. The reading may well not quite be perfect with regard to the initial characters, so that the display pattern in the drawing will flash for about 1 second before becoming stable. There are 255 synchronisation characters on the tape. These ASCII bytes take about 2.5 seconds to read. The Junior Computer is able to detect the beginning of a data block as soon as it has read an uninterrupted sequence of 10 synchronisation characters. Since there are 255 altogether, the Junior Computer has an excellent chance (in fact at least 20 possibilities) of making an unambiguous detection. The KIM, on the other hand, only has 100 synchronisation characters and so things are far more likely to go wrong here.

The third situation as shown in figure 3 occurs when the ID number specified by the operator has been found and loaded into the Junior Computer memory.

Before data on tape can begin to be read, that is, before the jump to the RDTAPE routine, an identification number must be specified. Even though a tape may feature up to 254 different data blocks, entering the number of one of them before the reading is enough for the computer to be able to trace it. There is also another method. If either ØØ or FF is introduced as an ID number, the computer will load the first data block that happens to arrive. If, however, ØØ is specified, the data block's number will be ignored and the block will be stored in memory at address SA

2

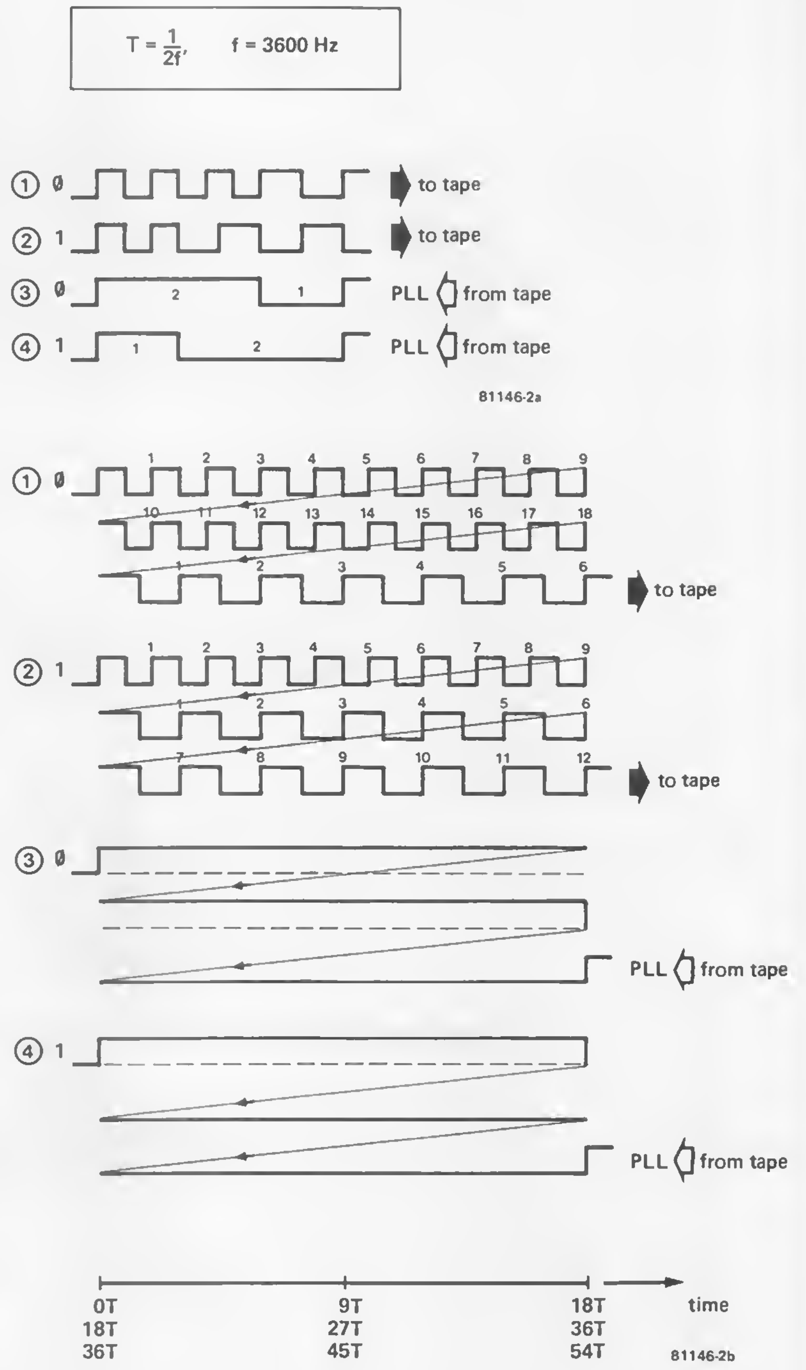


Figure 2. The signals that are transmitted on tape during the write operation correspond to a Ø bit ① (a) in the case of the Junior Computer and to a 1 bit ② in that of the KIM (b). The common time base at the bottom of figure 2 has been 'reset' twice to reduce the size of the drawings. The time base is split into 3600 Hz (T) half period units. The signals in ③ and ④ are produced when the tape is being read and the data has been processed by the PLL. After being inverted by the computer the signals are used to reconstruct the read data and store it in memory.

on the tape. If, on the other hand, FF is specified, both the data block number and the start address SA on the tape will be ignored. In that case, the data block will be stored in memory at an address selected by the operator at that particular moment.
This enables the data blocks to be moved around easily. The only consideration that needs to be taken into account is that whenever FF or ØØ is used as an ID number the 'first come, first serve' rule will be enforced, meaning that the first data block that happens to arrive in due form will be loaded in memory. In other words, the operator must know exactly which data block is to be transferred and where it is situated on the tape. The easiest solution is to use a cassette recorder which includes a counter.

DATA MANAGEMENT
The Tape Management program

The TAPE MANAGEMENT program (which we'll refer to as TM from now on) could also be called 'Tape Monitor'. It is designed to fulfil all the operator's wishes with respect to writing data from the Junior Computer onto cassette (= reading from the Junior Computer memory) and reading data from cassette into the Junior Computer (= writing into the Junior Computer memory). The program resides in a 2716 EPROM. Although this comprises more than 1024 bytes, it by no means occupies the full 2048 bytes, so that there is room for a few bytes which will come in handy later on.
TM ranges from Ø8ØØ to ØC7F. Nevertheless, its start address is not actually Ø8ØØ as might be expected, but Ø81Ø! Sometimes the TM is left by way of the editor (see figure 2) and sometimes it is exited from by depressing the RST key, after which the computer will be back in the standard monitor routine.
As soon as the computer is activated (AD Ø 8 1 Ø GO) the first situation in figure 4 will appear on the display. Then depressing the PAR key (= + key) will lead to the second situation in figure 4. Every time PAR is depressed, the next situation in the series will arise until, finally, PAR brings us back to the beginning again (drawing 1). As you have probably already guessed, 'PAR' stands for 'parameter'. This term is used to define the size of a particular data block and its whereabouts on cassette. As shown in figure 4, there are nine parameters altogether, one or several of which have to be specified (depending on which other function key(s) of the four that TM also acknowledges was/were selected). The parameters are as follows:

- ID (program or data block number)
- SAH
- SAL
- EAH
- EAL
- BEG(AD)H
- BEG(AD)L
- END(AD)H
- END(AD)L

It should be noted here that the high order address byte is specified first for once.
When the required parameter is shown on the display, two numeric keys are depressed, one after the other. The two key values (nibbles) move from right to left across the two left hand displays in exactly the same way as in the DA mode. After the start of TM, the nine parameters are reset (ØØ). This is precisely what is shown in figure 4. In this way, the PAR key allows data to be introduced which the Junior Computer needs to be able to successively carry out the read and/or write operation(s). This is very convenient for the operator, as he/she can now 'see' what is going on. If the more primitive system were to be used (AD Ø Ø E 2 DA X X + Y Y) the operator would not really know what was happening: ØØE2, is that BEGADL or BEGADH??...
N.B. Nine locations on page ØØ or 1A correspond to the nine initialisation parameters in figure 4. Locations 1A69 . . . 1A7F may not be overwritten during the cassette read operation!
What about those four function keys that were mentioned earlier? These are:
1. SAVE: this is a new name and a new function for the AD key. It enables data stored in the Junior Computer's memory to be saved by transferring them to cassette. Before the key is operated, the cassette is prepared for recording (depress 'rec' and 'play' simultaneously).
Again before this, however, an ID (neither ØØ nor FF) and the parameters SAH, SAL, EAH and EAL have to be indicated for the data block that is to be recorded. Take care! EAH and EAL form an address which is located one place behind the last address of the data block concerned. Thus, if the end of data address is Ø3FF, EAH = Ø4 and EAL = ØØ.
When the SAVE key is operated it calls the routine DUMP. The red LED D5 is lit, but the six displays will remain unlit. Once the data block has been recorded, the Junior Computer will announce this by displaying 'ID XX' (first drawing in figure 4), where X is instead of ØØ, being the number of the data block which has just been copied on cassette.
N.B. Operators will find it highly useful to jot down the ID, SA and EA, as well as the value displayed by the cassette recorder counter, on a piece of paper.
2. GET: this is a new function for the PC key. When it is operated the Junior Computer reads a certain data block stored on cassette and memorizes it. The cassette recorder will have been put on 'play' beforehand. As before, it is enough to specify the ID number. Only ID numbers included between Ø1 and FE will be stored on the cassette. If, before depressing GET, ØØ is introduced as an ID number, the Junior Computer will store the first data block it meets in memory, without taking any notice of its ID. Instead, the data block's SAH and SAL will determine where the block is to be placed in memory.
If, on the other hand, FF is introduced as an ID number before GET is operated, the Junior Computer will again

3

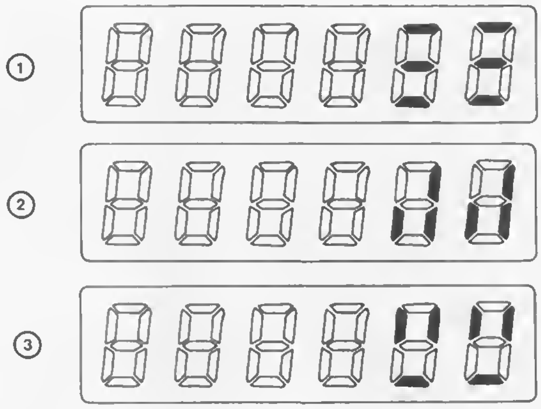


Figure 3. When a data transmission (data block) is being searched, detected end read on cassette, information will be shown on the two right hand displays of the Junior Computer. This is not the case in the KIM computer.

store the first data block it comes across in memory, regardless of its ID, but now the block will be stored in an address range specified by the operator and the SAH and SAL cassette will be ignored.
Depressing the GET key calls routine RDTAPE. Then the green LED D4 will light and one of the situations indicated in figure 3 will be shown on the display. Once the block is loaded, the Junior Computer will inform the operator by displaying 'ID XX' (situation 1 in figure 4, where XX is a number between ØØ and FF).
N.B. When the data block is loaded using FF as its ID, only the contents of ID and SAH/SAL will be correct. So don't expect to find the end of block address in EAH/EAL!

In addition, something else happens: the start address (contents of SAH/SAL) on display will constitute the end of data block address. This allows edited programs to be placed side by side without loss of space (the EOF characters are suppressed). Bearing this in mind, if any data blocks are to be placed side by side using ID = FF, EAL will have to be modified (incremented by 1) and, if necessary, so will EAH, before a new data block is loaded. This type of storage will obviously be necessary when a program is to be reconstructed from various sections dispersed here and there on the cassette. In that case, it is very important to know 'what's where'!

3. EDIT: this attributes a new function to the DA key. In actual fact, the function concerned is not entirely new, as activating EDIT is equivalent to operating AD 1 C B 5 GO, leading to a cold start of the editor. Prior to this, BEGAD and ENDAD are introduced by means of the PAR key (operate PAR until BEG(AD)H is found, enter the corresponding data, operate PAR and enter suitable data).

As you know, a cold start in the editor causes '77' to appear on the left hand displays. This is exactly what happens when EDIT is operated as well, so that it may well seem to be superfluous. Is is really necessary? The answer is yes, as will be understood from the following.

4. SEF: this is a new function attributed to GO. It has a special task to fulfil just before TM is started. SEF stands for Save Edited File; in other words, a data block that is *not completely edited*, and therefore *not yet assembled*, is transferred to cassette. Provided the EDIT key is depressed beforehand (cold start editor), all the program data from BEGAD onwards (like ENDAD, BEGAD will have been entered prior to depressing EDIT) up to the address to which the variable pointer CEND is pointing, will be copied on tape in the form of a single data block during the DUMP routine. For such a block to be transferred by means of the SEF key, it is absolutely

4

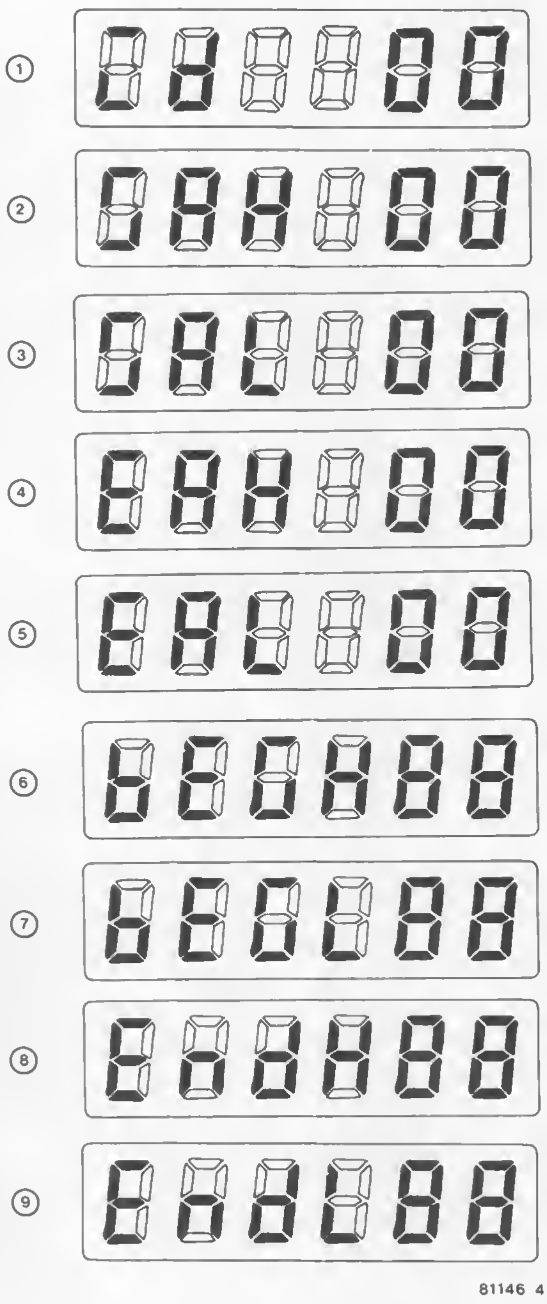


Figure 4. The nine parameters which play a decisive role when data is being transferred from or to the cassette, can all be brought into view during the TAPE MANAGEMENT program with the aid of the PAR = + key, so that data may then be keyed in (shown on two right hand displays).

imperative that BEGAD and CEND are defined, which of course is only possible once the program has been edited! Thus, before SEF is operated to tape an edited program on cassette, the editor must be left by way of the monitor and the computer then jumps to TM (during this time, the Junior Computer must under no circumstances be switched off — BEGAD and CEND are stored in RAM in page ØØ!!):
RST Ø 8 1 Ø GO
enter the ID
and operate SEF.
It is during the DUMP routine that data is written on tape. The display remains unlit until SEF is operated, but the red LED is lit. Once all the data has been transferred the first instruction of the edited program that has just been taped

will appear on the display. This is because after DUMP TM ensures a warm start entry into the editor. Previously, of course, a program number had to be entered and as for the start address, this is equal to BEGAD, the end address (the end of data block address plus one) being equal to CEND.

How can we benefit from SEF? This function allows us to store incomplete (non assembled) programs of any length by 'putting them on ice' so to speak. All that has to be done to fetch such data blocks is to depress the GET key and then activate the editor by means of a warm start entry. Let us see how this is done. The variable end of address pointer CEND is pointing at the first vacant location in memory at the end of the data block. Straight after the EOF character, in other words. If *your* memory is failing you on this point, it is a good idea to take another look at chapter 8 in Book 2. Since it is the CEND pointer that acts as the EA during an SEF operation, the last data to be read in the block will be 77, the EOF character. The operator does not have to enter BEGAD = SA and CEND = EA before depressing SEF, as this is done automatically by the machine. What must be specified, however, is the program ID number, which is as well to write down! Why? Well, because after the data block is reread:

RST or AD Ø 8 1 Ø GO X Y (XY = program number) the computer must be prepared to jump to the editor.

First of all, RST (return from TM to monitor) is depressed. Then the contents of BEGAD (L = ØØE2, H = ØØE3) are made to equal that of BEGAD = SA as was noted during the data block recording. After this the contents of CEND (L = ØØE8, H = ØØE9) are made to equal CEND = EA which will also have been noted previously. All that remains is to make CURAD's contents equal BEGAD's and then the computer may proceed with the warm start editor entry:

AD 1 C C A GO.

Since CURAD now equals BEGAD, the first instruction will appear on the display.

N.B.: The editor should *never* be activated by means of a *cold* start entry in this particular instance, by depressing EDIT, say. This would inadvertently cause 77 to appear instead of the first instruction, which would really put the cat among the pigeons.

N.B.: When several blocks of edited data are being read on tape and ID = FF, the value of CEND will be equivalent to the address of the last data block to be read.

An edited and taped data block (or several) may be re-addressed by re-reading it (them) with FF as the ID. It goes without saying that in that case the CEND and CURAD parameters must be adjusted accordingly.

```
(RST 1 Ø Ø Ø GO)

(CTRL+DEL=RUB)
JUNIOR

1A7E(SP)
1A7E C4 CF.
1A7F 09 14.
1A80 80 100(SP)
0100 5D 18.
0101 1D A9.
0102 1C 13.
0103 3D 69.
0104 3C 08.
0105 3C .
0106 2C +
0107 AC -
0106 2C -
0105 00 -
0104 08 -
0103 69 -
0102 13 -
0101 A9 -
0100 18 L
ACC: C8
Y  : CB
X  : CA
PC : F33C
SP : 01FF
PR : 00000100
     NV BDIZC 100(SP) (STEP:OFF)
0100 18 R
0107 AC L
ACC: 1B
Y  : CB
X  : CA
PC : 0107
SP : 01FF
PR : 00110100
     NV BDIZC F3(SP)
00F3 1B 100(SP) (STEP:ON)
0100 18 R
0101 A9 L
ACC: 1B
Y  : CB
X  : CA
PC : 0101
SP : 01FF
PR : 00100100
     NV BDIZC P
0101 A9 R
0103 69 L
ACC: 13
Y  : CB
X  : CA
PC : 0103
SP : 01FF
PR : 00100100
     NV BDIZC P
0103 69 R
0105 00 L
ACC: 1B
Y  : CB
X  : CA
PC : 0105
SP : 01FF
PR : 00100100
     NV BDIZC P
0105 00 R
0107 AC L
ACC: 1B
Y  : CB
X  : CA
PC : 0107
SP : 01FF
PR : 00110100   (STEP:OFF!)
     NV BDIZC M
HEXDUMP: 100,105
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0100: 18 A9 13 69 08 00

JUNIOR

S11,100,106
READY
```

The Printer Monitor program

How the computer writes home to the operator

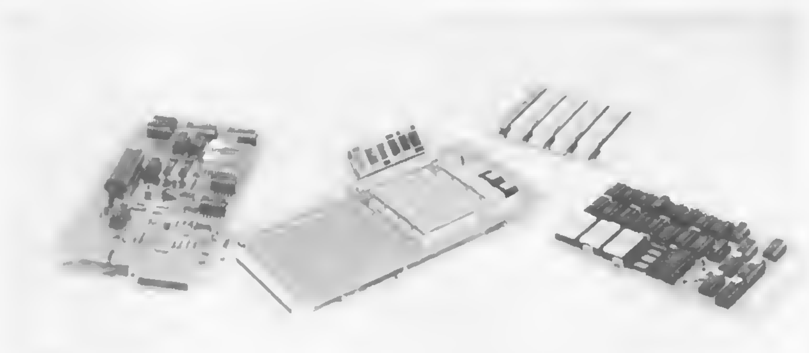
The PRINTER MONITOR program (which will be called PM from now on — no relation to the resident at no. 10!) will be seen to consume yards of paper when a printer is connected to the computer. To describe it in detail would take a few reams as well! The program is stored inside a 2716 EPROM and extends from address 1ØØØ to 14F3. Again, the EPROM has enough room left for other resident programs.

The PM features the following standard function keys: AD, DA, + and GO. This time, however, data will not be overwritten by other data on the six Junior Computer displays, but we will end up with a complete case history of everything we do. If a printer is used, the paper may roll on indefinitely and all the information will remain permanently available, which is certainly not the case with a video terminal, where the memory and display capacity is of course fairly limited.

The PM is started by way of the monitor:

AD 1 Ø Ø Ø GO

Depressing the Elekterminal's RUB key (CTRL or DEL in other devices), will cause the Junior Computer to answer by displaying the word 'JUNIOR'. The ASCII keyboard keys Ø . . . 9 and A . . . F may then be used to enter a work address. Insignificant zeros may be omitted: thus, '200' stands for 'Ø200'. The work address appears with the contents of the corresponding memory location, once the SP (SPACE) key has been operated. If data is to be entered at this address, the first two keys (Ø . . . F) are pressed and then the "." key (full stop). The information is stored in the Junior Computer's memory which subsequently displays the following address and its contents. The same procedure is repeated.



Function keys

In addition to the auxiliary keys, RUB, CR etc., there are ten function keys:

1. The "—" key: this enables the address immediately preceding the work address to be printed; in other words the work address is decremented.
2. The "+" key: this increments the work address in the same way as the + key on the standard Junior Computer keyboard. The new address is shown with its corresponding memory contents.
3. The SPACE key: the specified work address appears together with its contents. This key is similar in every way to the AD key on the standard keyboard.
4. The "." key: the last data to be entered is stored at the work address. This operates just like the DA key in the standard Junior Computer.
5. The R key: "R" stands for "run" and operates like the GO key; the program is started from the last address to be printed (= work address).
6. The L key: "L" stands for "list". When this key is operated the contents of all the internal registers belonging to the 6502 μP, including ACC, Y, X, PC, SP and P, are listed. The P register is represented by eight bits, each one of which has an identification letter: N, V, (space), B, D, I, Z and C.
7. The P key: "P" stands for "print" and is depressed to show the PC (program counter) contents, as in the step by step mode just before the program is left (after an instruction was executed) to go to the PM. Thus, similar to the PC standard key, the P key makes sure the following instruction is prepared (depress R). Step by step programming is only possible in the PM when S24 is "ON" (the LED of the GO key will then be lit) and provided the circuit around IC10 on the main board is correct (see figure 1b in the 'Junior Cookbook' article in the April issue).
8. The M key: When this key is depressed, the text 'HEXDUMP' is printed. Then an address is entered (without the insignificant zeros), the "," (comma) key is depressed and a second address is entered. Depressing CR will cause the hexdump between two specified addresses to be listed. At the beginning of every row of 16 data the address of the first data in the line will be shown. Each data column is headed by a figure between Ø and F, enabling a certain data address to be found. The last line in the hexdump is not necessarily complete, as the number of addresses does not have to be a multiple factor of 16.
9. The G key: "G" stands for "GET". Depressing this, then the program number required (ID) and finally the CR key, will tell the Junior Computer to search the data block corresponding to the ID specified on the cassette tape, and copy it into memory (provided the recorder is on 'play'). Once the read operation has been completed, the word 'READY' will be printed. This means all has gone well. If the program number was specified as ØØ, the first data block the Junior Computer comes across will be read and stored in memory. If the ID was FF, "SA" will appear and this address will have to be entered, after which the first suitable data block will be searched and stored in the computer's memory at an address specified by the operator.
10. The S key: this allows a data block to be written on tape. This is what happens: The S key is depressed, then the required program number is entered, the "," key is depressed and the start address SA is entered. Next the "," key is operated again, the end address SA is entered (one address behind the last data block), the recorder is set on 'record' and is started. Finally, the CR key is operated. Once everything is correctly recorded, the Junior Computer will let you know by printing "READY".

That brings us to the end of our survey, and we would like to conclude by adding a few helpful hints.

Operators know how to use the numeric keys (ASCII Ø . . . 9 and A . . . F) and the others, such as "." and SPACE, to enter a work address, or to modify data. As soon as a function key is executed, the corresponding data buffers are reset. After this, the SPACE key is operated and the work address becomes ØØØØ. If "." is pressed at a certain work address, the contents of that particular address will become ØØ.

The PM program automatically specifies the NMI vector. As a rule, NMIL = CF (address 1A7A) and NMIH = 14 (address 1A7B). This relates to the step by step execution of a program. If a program ends in a BRK instruction, a direct jump may be made to PM, provided the IRQ vector is correctly positioned. As a rule, IRQL = CF)address 1A7E) and IRQH = 14 (address 1A7F). Data relating to the vector may be entered either before or after the PM has started.

After the start of a program (key R, with work address = start address) ending in a BRK instruction, the Junior Computer will again display or print the address and its contents. This address will be two addresses after the one containing the BRK instruction (provided the IRQ vector indicates 14CF). In the step by step mode, the Junior Computer will report back by displaying or printing the address containing the opcode of the following instruction. ■

reading bar codes

From bars to bytes

Simple computer programs are already being printed in bar code form for calculators. Naturally there must be a reliable means of reading them at high speed. This requires two essential items: a good quality light pen and an intelligent measuring system.

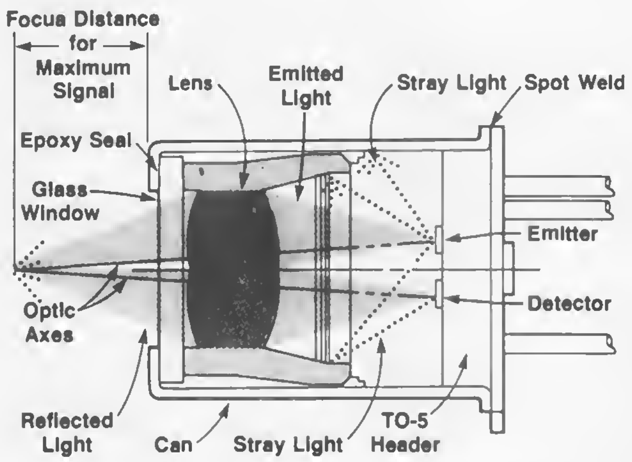
One bar per bit; eight bits per byte; sixteen bytes per printed line and 64 lines per Kbyte . . . that's more than eight thousand bars spread across several pages, which will all have to be translated into logic ones and zeros without errors.

The light pen

Since black bars reflect less light than white spaces, it would seem a simple matter to read them by scanning the pattern with a photo transistor. A

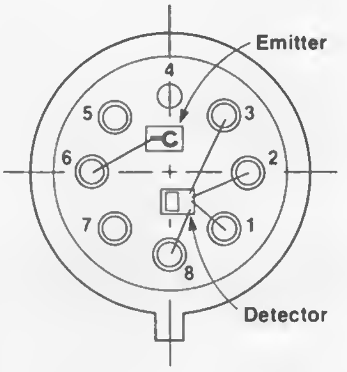
square-wave signal would then be produced at the output. Since the light in a room is not very reliable it might be a good idea to mount a lamp next to the transistor, making sure its light falls onto the bars and not on the photo transistor. This can be done by inserting a panel between them. All in all, a simple job? Think again . . .
Bars may be very narrow. For the sensor to be able to differentiate between a 0.25 mm zero line and a logic one of twice the width, it will need a measuring accuracy of about 0.1 mm! Not so easy to construct after all . . . Never mind, there are various suitable types available ready-made on the market. Hewlett-Packard have manufactured the HEDS-1000 light sensor shown in figure 1. The light source consists of a small LED and the sensor itself is made up of a photodiode and a transistor fraternally combined into a single chip. The circuit diagram is drawn, in a slightly unusual way, in figure 2. This figure also indicates how to construct an effective sensor by adding three resistors.
Obviously the output signal will be too small to drive TTL gates. Not only are the resistor values very high, but the signal level is far too low. It will have to be amplified. In addition, something will have to be done about cleaning up the output, for the original bars may be slightly ragged and not quite black. As for the spaces, they may well be smudged here and there. Even when there is an ideal, clear black-white transition, the output signal will rise gradually as the light spot is not infinitely small. At its best the output may resemble that shown in figure 3a, but more often than not it will look like figure 3b.
There are various techniques which can help to locate the light-to-dark and dark-to-light transitions. The signal may be differentiated for instance, so that the high-low and low-high transitions are retrieved from the signal. This does however, have a number of serious disadvantages.
This system is highly sensitive to interference (both printing errors and electrical interference) and what is more the light pen must be moved along the code lines at a correct and fairly constant speed.
Another possibility would be to amplify the signal considerably so that it starts to clip. This will only work if the signal is kept at a fairly constant level.
Yet another possibility would be to detect the positive and negative peaks of the signal and to assume a 'zero-crossing' when the signal is half way between the two extremes. This requires high-speed peak detectors. Taking a signal such as the one shown in figure 3b, the two reference levels (signal maximum and minimum) will have to be calculated at every individual peak for such a fluctuating signal to be processed properly. In many codes it is important that the first bar is measured correctly. At that particular moment the low-level detector (corresponding to black) may well have a random value, so that it will be pure coincidence if the first white-to-black transition happens to be read correctly.
Hewlett-Packard provide a complete light pen, the HEDS-3000 which contains a 'finishing touch' circuit. The design is reliable, low-cost and uses a minimum of easily obtainable components. Just the home-builder's cup of tea!

1a



81145-1a

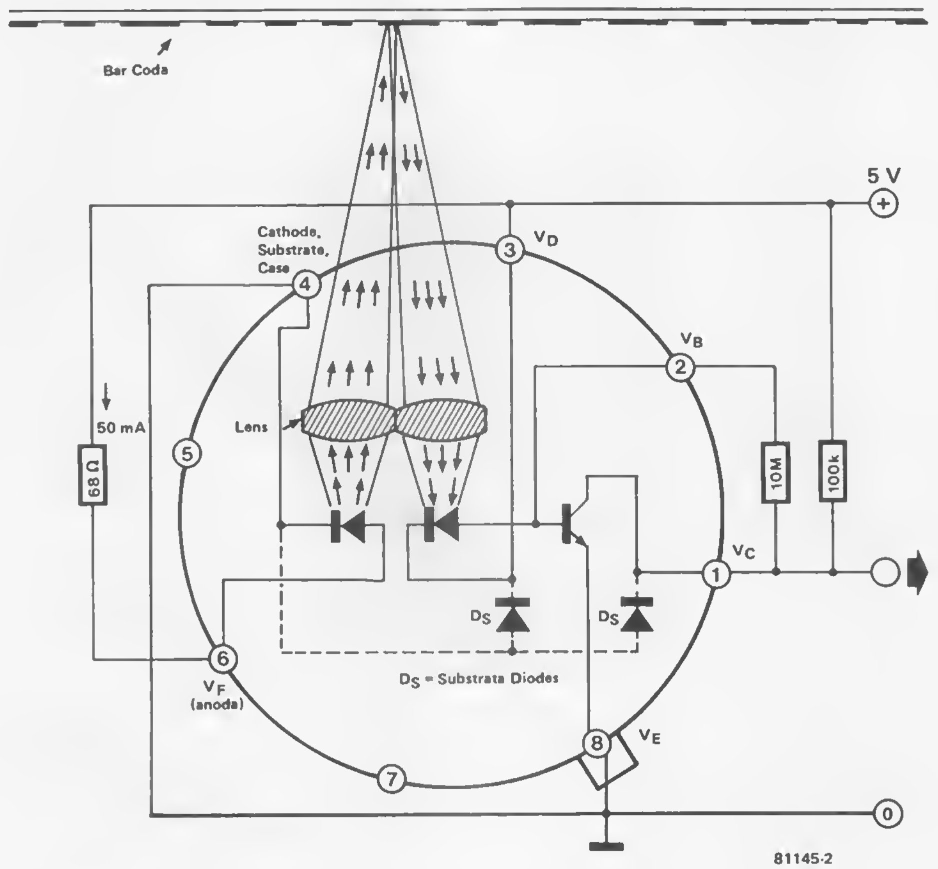
1b



81145-1b

Figure 1. Cross-section and bottom view of the HEDS-1000. This is an ideal sensor for light pens.

2



81145-2

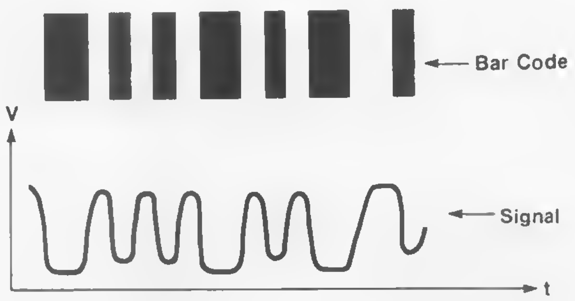
Figure 2. The circuit diagram of the HEDS-1000. The figure also includes the pinning. Adding three external resistors is enough to make a very simple light pen.

The circuit is shown in figure 4 and consists of three main sections: the amplifier, the signal processor and the output buffer. The amplifier has to boost the output current of the sensor (about 100 mA) to a few volts. This signal is fed to a simple peak detector built around D3, D4 and C1. C1's voltage is compared with the original signal. Operation is explained in figure 5. Initially the pen is pointed at the blank space on the page, to the left of the first bar. A1's output will be high at that moment and the voltage across C1 is about 0.7 V lower than this. Then the pen reaches the black bar and the output voltage of A1 drops. At a certain moment this voltage will drop below that of C1. The output of A2 will become high, T1 will conduct, T2 will stop conducting and the output become high. If the output voltage of A1 drops any further, D4 will start to conduct at a certain moment. The voltage across C1 now drops to about 0.7 V above the signal level. This negative peak reference is then used to detect when the output voltage of A1 starts to rise again. The output of A2 (and therefore the logic output T2 as well) will then go low.
There are a number of considerations to take into account. For one thing, the output signal of A1 must be high enough to allow D3 and D4 to conduct in turn, but in practice the results are satisfactory. The width-error for the first bar is given as 0.1 mm and for the following bars this will be 0.05 mm.

The measuring system

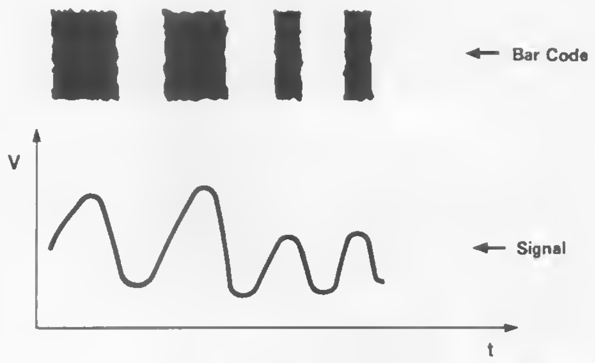
A light pen is still a far cry from a bar code reader. All it does is to convert the black bars into a kind of PDM signal at a TTL level. The system has yet to include error detection and decoding, which will involve quite a bit more electronics. Plessey uses a double Eurocard to mount the entire code converter a very good one, admittedly.
The HP version, however, is preferable to Plessey's for our application, as it was specifically designed for software distribution. In other words, it is used in combination with an existing computer system, and it makes sense to let the computer do most of the work. Furthermore the code it entails is much simpler, as the bars merely represent zeros and ones and these are the actual data. This means the circuit can be fairly straightforward. The converter connected between the light pen and the computer now consists of only two ICs which are incorporated in the connector at the end of the light pen cable! As you might have guessed, these are not common or garden ICs. One of them is in fact a microprocessor-based chip that was specially designed for this application and the other is a 4K word ROM.
The interface converts the electrical signal of the light pen into binary (a narrow bar corresponds to a logic zero and a wide one to a logic one). The

3a



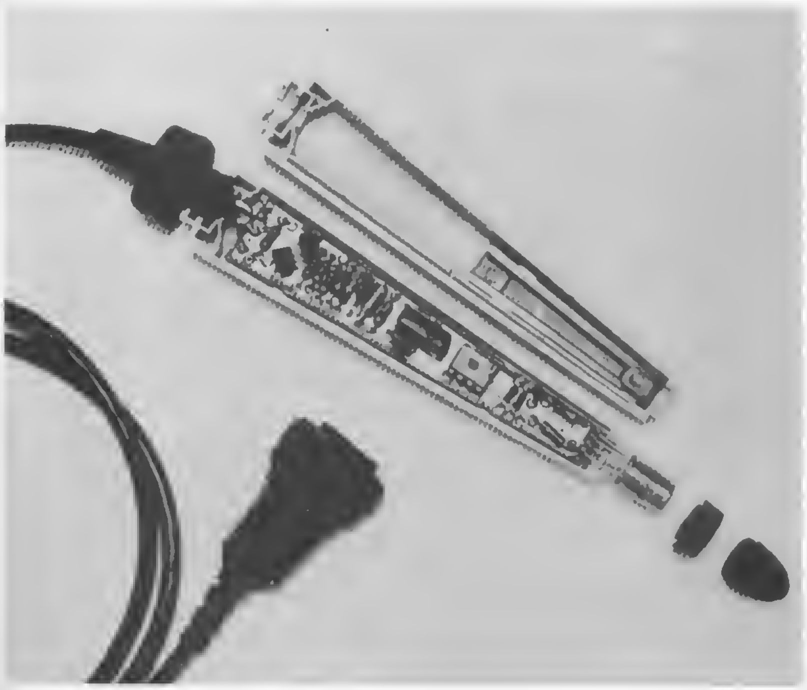
81145-3a

3b



81145-3b

Figure 3. Two examples of the type of signals that the sensor is likely to produce. Figure 3a shows a fairly good print sample and figure 3b illustrates what can go wrong.



4

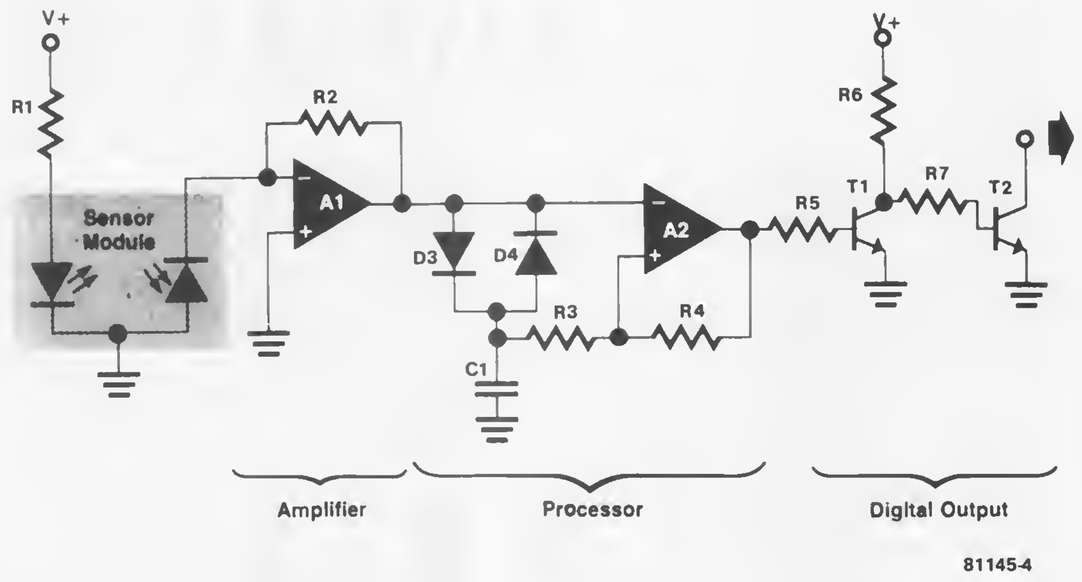


Figure 4. The circuit diagram of the complete light pen manufactured by HP, the HEDS-3000. Only the power supply decoupling (two capacitors) is not included here.

5

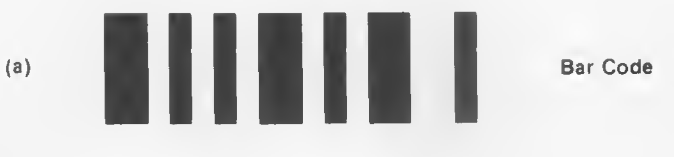
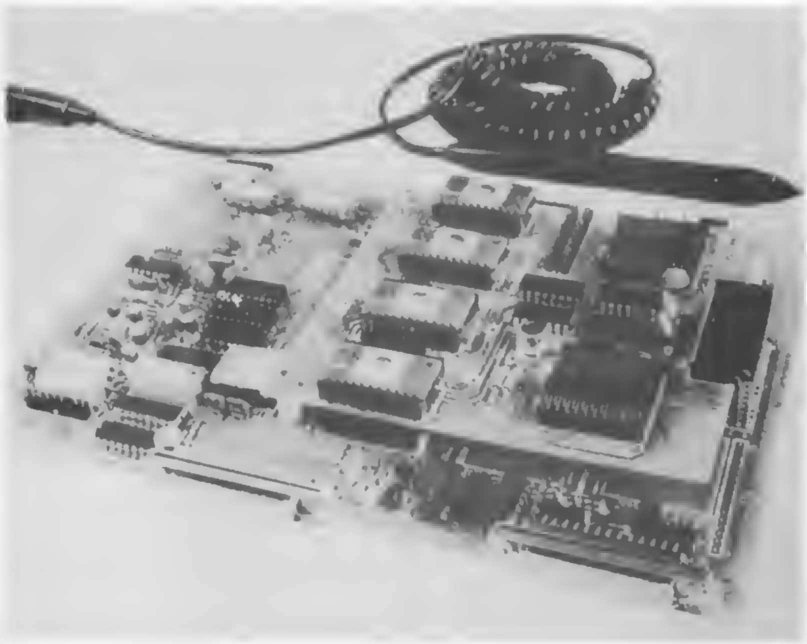


Figure 5. Signal processing in the HEDS-3000. When the pen is moved along the bar code (a), the signal is amplified as shown in figure 5b. In addition the reference signal is derived from the bar code by way of two diodes and a capacitor. When the two signals are compared this leads to the output signal.

data is then split into groups of eight bits (bytes) and the bytes are stored in a working storage memory, after which the interface 'consults' the microprocessor in the HP 41C by way of the bus lines until the bytes have been correctly transferred to the calculator.
How does the system work? Well, everything is centred around the bar code itself. In the case of the one used by HP, the narrow bars and the space between two bars are equally wide. A logic one is represented as a bar of twice that width. It was seen before that the output of the light pen is low when it scans bars and high when it passes across blank spaces.
Decoding is achieved by counting the HP 41C's clock pulses (360 kHz). This is first done whilst the signal of the light pen is low (bar width) and subsequently during the 'high' period (space).
The average is calculated from the two values for the number of the clock pulses corresponding to a space or to a narrow bar. Next the result is compared with that of the following bar. If more than one and a half times the reference number of clock pulses were counted, the new bar will be acknowledged as a logic one; if there were fewer, it must be a logic zero. Let's look at an example. First a bar is measured and found to be 1100 pulses wide. Next there is a space of 900 pulses. This gives an average of 1000 pulses. The reference value will therefore be 1½ x 1000 ≙ 1500 pulses. The next bar turns out to be 1600 pulses wide; as this is more than 1500 it will be a logic one. Now a new space follows, 700 pulses wide. The 'unit



width' of the wide bar corresponded to 800 pulses (its total, double width was measured as 1600), so the new reference value will be 750 pulses. In other words, the light pen is picking up speed!

All this still fails to guarantee absolute reliability. A fairly elaborate program is also involved, as shown in figure 6. As soon as the light pen touches the paper, the counter starts to run. While a maximum value is reached ($\geqslant 2^{14}$ clock pulses) the system will enter a delay loop until the first bar appears. Its width is then 'counted' along with that of the following space. These two pieces of information lead to the average unit width used to judge the next bar. If the second bar is a narrow one, the system knows the light pen is scanning the bars from left to right, a wide bar will indicate the opposite.

After this the actual bit by bit decoding may take place. Groups of 8 bits are collected as bytes and stored in a 16 byte buffer. There are two such buffers available so that the processor can read one buffer while the pen is

6

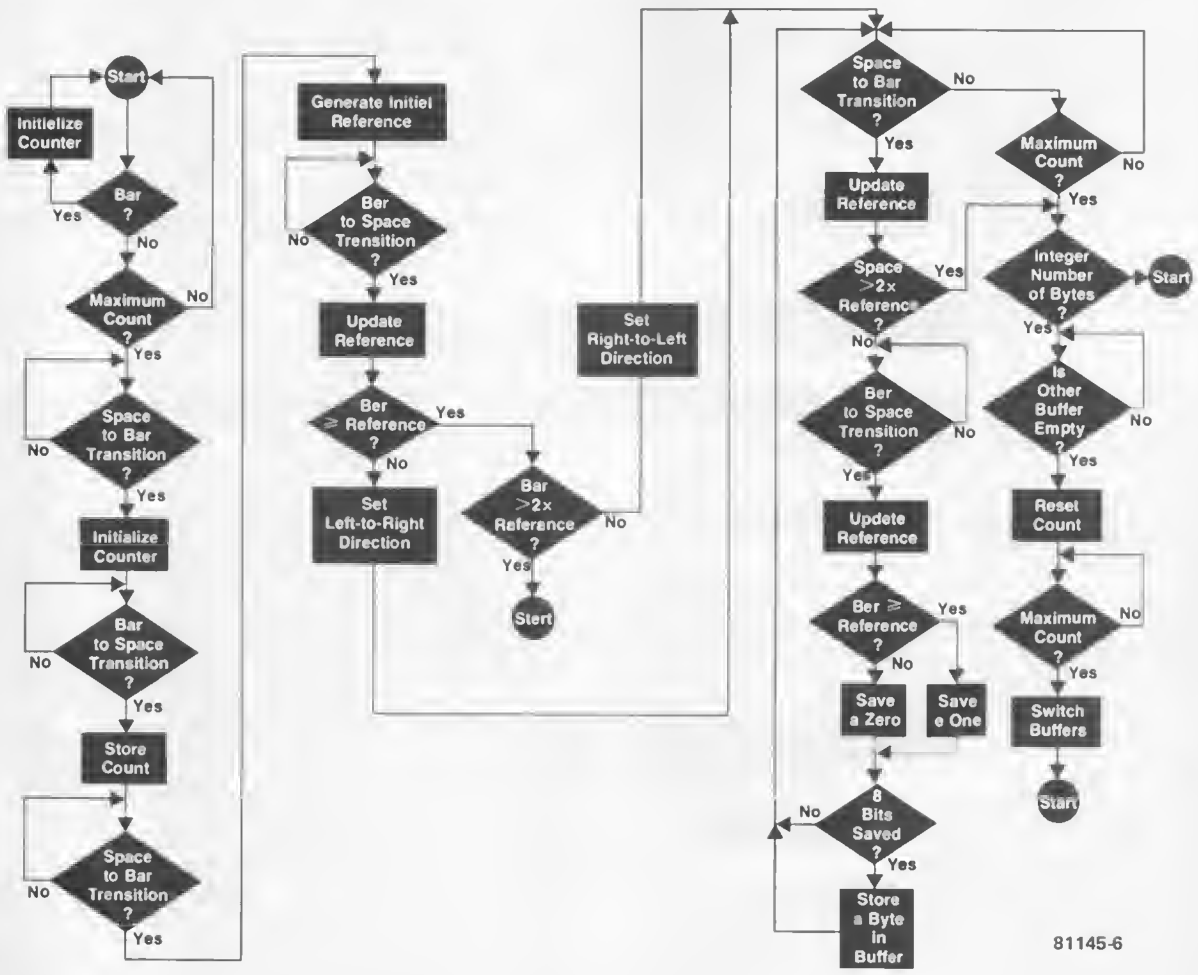


Figure 6. A flow chart showing the decoding procedure in the HP-system.

7

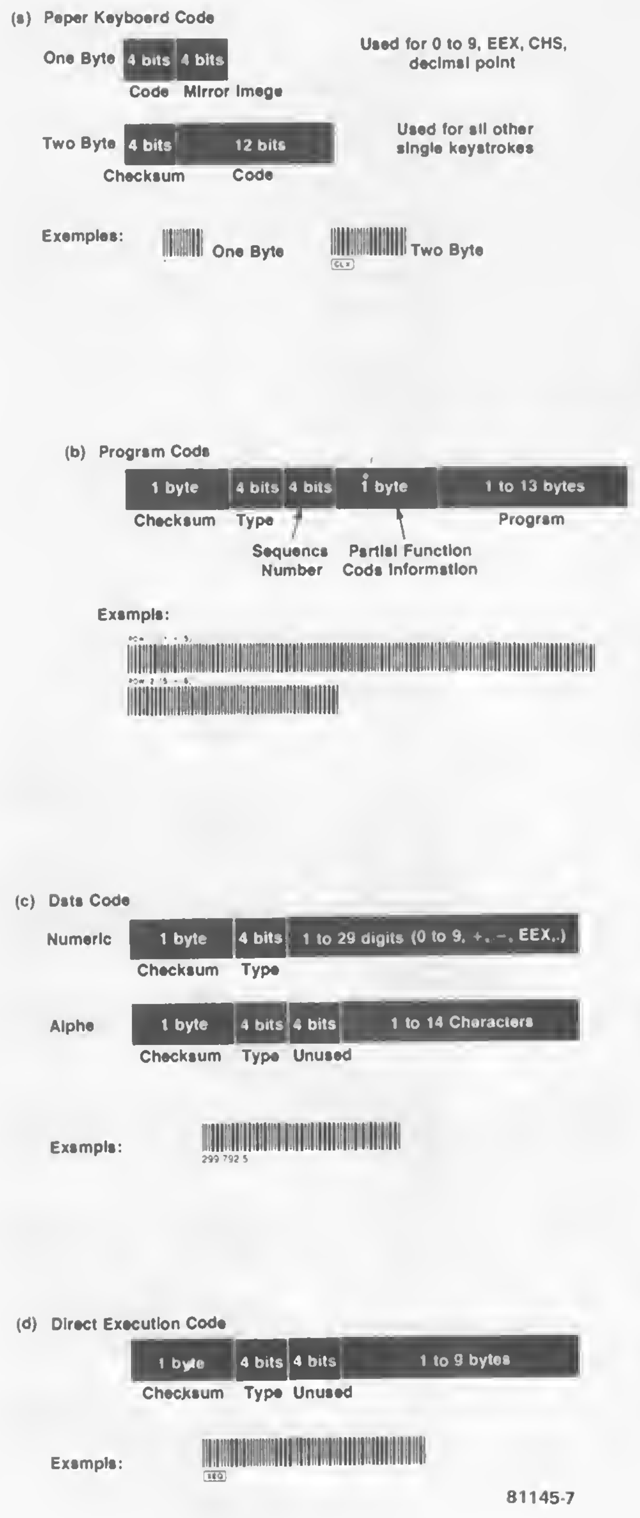


Figure 7. HP uses bar code for four different applications. The 'program code' and 'data code' uses are of course compatible to other (microcomputer) systems.

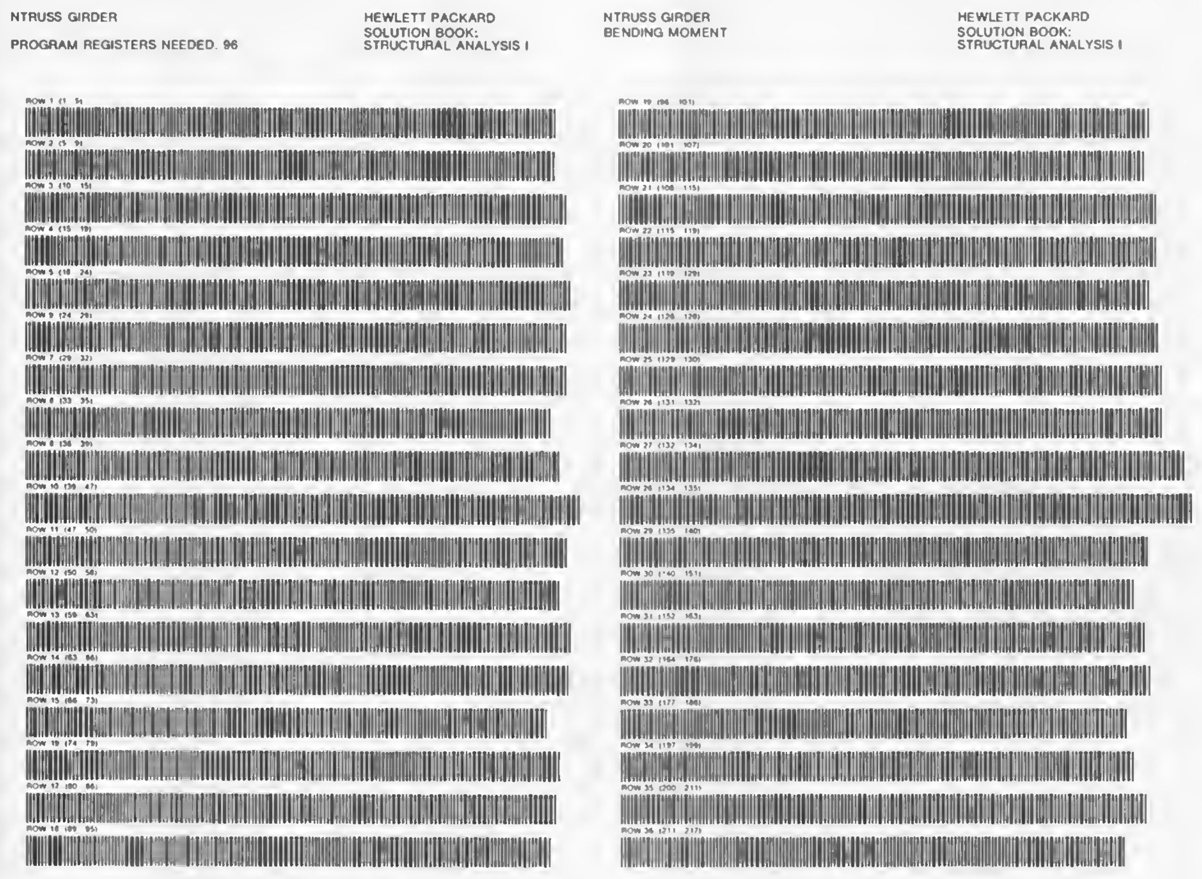
simultaneously writing into the other. Various checks are carried out during the decoding. If at the end of the road the last byte turns out to be incomplete (less than eight bits), something must have gone wrong. Furthermore, the spaces may not be too large, as if they have twice the reference width (or three times their 'official' width), this will be interpreted as the end of the line. The same will also happen if the maximum count (2^{14} clock pulses) is exceeded. Seeing as the spaces between the bars are just as wide as the narrow bars, this establishes the maximum bar width: when the light pen is moving at its lowest speed (about 7.5 cm per second) the bars may not exceed 3 mm.

However, since the whole idea is to squeeze as many bars onto a page as possible, 3 mm is much wider than necessary. It is more interesting to find out the minimum width. This depends on the time necessary to decode a bar, calculate the next reference value and, if required, store a group of 8 bits in the memory. All this takes 76 clock pulses and as the highest speed of the light pen is 76 cm per second this gives a minimum bar width of 0.17 mm. That's cutting it rather fine! The light pen will have to be good, not to mention the printer.

From rough bytes to ready data

The system as described above takes care of the entire bar code reading procedure up to the point where a maximum of 16 bytes are stored in an intermediate memory. From then on it will be up to the computer (in this case the HP-41C) to do something with the data.

8


NTRUSS GIRDER

PROGRAM REGISTERS NEEDED. 96

HEWLETT PACKARD
SOLUTION BOOK:
STRUCTURAL ANALYSIS I

ROW 1 (1 5)
ROW 2 (5 9)
ROW 3 (10 15)
ROW 4 (15 19)
ROW 5 (18 24)
ROW 9 (24 29)
ROW 7 (29 32)
ROW 8 (33 35)
ROW 8 (36 39)
ROW 10 (39 47)
ROW 11 (47 50)
ROW 12 (50 58)
ROW 13 (59 63)
ROW 14 (63 66)
ROW 15 (66 73)
ROW 19 (74 79)
ROW 12 (80 86)
ROW 18 (89 95)

NTRUSS GIRDER
BENDING MOMENT

HEWLETT PACKARD
SOLUTION BOOK:
STRUCTURAL ANALYSIS I

ROW 19 (96 101)
ROW 20 (101 107)
ROW 21 (108 115)
ROW 22 (115 119)
ROW 23 (119 129)
ROW 24 (128 128)
ROW 25 (129 130)
ROW 26 (131 132)
ROW 27 (132 134)
ROW 28 (134 135)
ROW 29 (135 140)
ROW 30 (140 151)
ROW 31 (152 163)
ROW 32 (164 176)
ROW 33 (177 186)
ROW 34 (197 199)
ROW 35 (200 211)
ROW 36 (211 217)

81145-8

Figure 8. Two sample pages from a bar code book. Their actual size is the same as Elektor magazine pages!

This calculator can use bar codes in four ways, as indicated in figure 7. The first method is unusual for a start: 'paper keyboard'. This involves lines of one or two bytes that represent a certain key function. The advantage is that one bar code line will determine a complete function, even if this can only be called up on the 'real' keyboard by depressing a whole string of keys.

A similar, if more elaborate, application is 'direct execution'. This is where a complete series of key strokes are imitated in one go: STO 12, for example. The final and by far the most interesting applications are the 'data' and 'program' codes. In both cases there is an 8 bit checksum at the beginning of each line. This is the result of adding up all the bytes (during 'program code') belonging to both the present line and the one(s) preceding it. This is a vital way to avoid errors. Then four bits follow to indicate the type of information that is concerned in this instance (either program or data). For program code another four bits appear to indicate the number of the line that is being read (to detect whether a line has been missed, or read twice by mistake) and a further 8 bits referring to any multiple byte functions that might have started on the last line or will run on to the next. Subsequently, another thirteen bytes (maximum) of 'real' program information may be transferred, either 29 digits or 14 different characters.

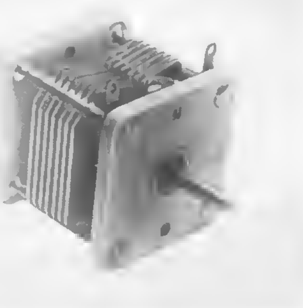
Hewlett-Packard manages to print about 18 lines on one page, as can be seen in figure 8. It looks rather like a lawn, doesn't it?! Still it saves hours of depressing keys, as the rows can be scanned with a light pen in a matter of minutes.

Hewlett-Packard Journal, January 1981

market

Differential sensor capacitor

A differential sensor capacitor developed as an edaptable mechanical element that can be designed into transducer heads of many different models, has been introduced by Jackson Brothers (London) Ltd.
The sensor, designated the C11K, has virtually unlimited life. It can be used to measure angular or rotational displacements directly. By measuring rate of change of capacitance it can also be used to measure rotational speed.



The C11K has air dielectric, silver-plated and soldered brass vane-packs and long-life, low-torque bearings.

Jackson Brothers (London) Ltd.,
Kingsway,
Waddon,
Croydon CR9 4 DG,
Telephone: 01-681 2754.

(1959 M)

EPROM programming system

A comprehensive and powerful EPROM programming system for use with the PET microcomputer has been developed by Davidson-Richards Limited. The system comprises a programming unit which fits directly to the user and cassette ports of the PET, and a disk containing the software. The programmer is provided with a socket to accommodate a 2 K or 4 K EPROM and a switch to indicate its size. The machine code program gives rapid user response, and fast writing speed.
Data can be read into the system work area from a disk file, the PET's own RAM or ROM or an EPROM mounted in the programmer socket. An easy to follow screen display with prompts enables the user to manipulate the data, and choose either of two methods of display. A listing in address order with code in hex, disassembly into 6502 mnemonics and operands and ASCII equivalents, or a straight hex dump.
Relocation is achieved simply by typing in a new start address, and individual bytes are modified by overtyping existing data. After modification, the code can be moved to a storage area to allow a further block to be prepared or written to an EPROM. After

writing, the contents of the EPROM can be verified against the memory image, and any differences displayed with the address and both sets of data.
The disassembly feature enables an instant check on input, preventing time consuming debugging for typing errors. EPROM's for use in other systems (e.g. Z80) can be coded and written, but the disassembly feature cannot be used.
A 16 K or 32 K Model Pet with Version 3.0 or 4.0 Basic Rom's is required for this system. A tape version is elso available, but this does not allow data to be read from its files into the work area.

Davidson-Richards Ltd.,
14 Duffield Road,
Derby DE1 3BB.
Telephone: (0332) 366803

(1971 M)

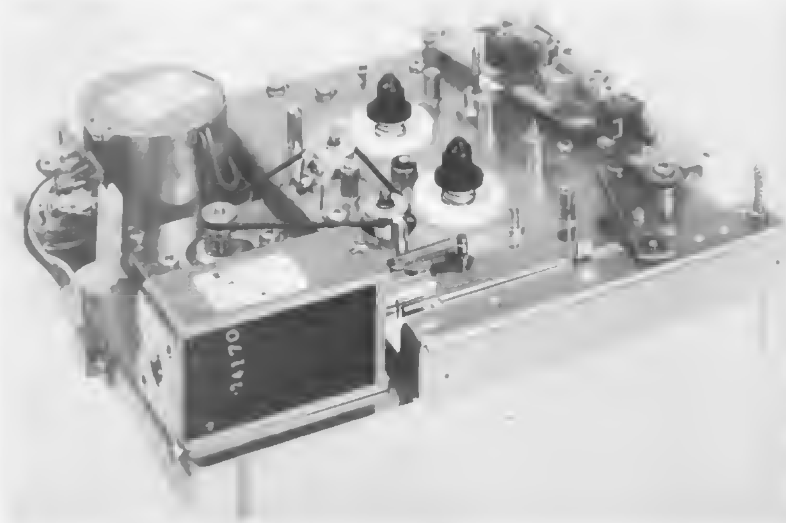
High quality cassette deck

A new, low cost, high quality cassette tape deck, the C1000, has been introduced by Monolith Electronics. Aimed at the home mini-computer enthusiast, the C 1000 is a development of Monolith's proven C 2000 series tape transports which are widely used in industrial process control, data logging and computer applications. It can be operated either remotely or locally via the control circuitry.
The C 1000 is designed around the international compact cassette and consists of two motors (capstan and spooling), head plate and solenoid. It operates from a nominal 12 V d.c. supply.
The basic control circuitry allows remote or local selection of fast forward, fast rewind, record and play. A Hall effect sensor and an opto end of tape sensor are optional extras to the control circuit end are supplied separately.
The Hall effect sensor can serve two functions viz. detection of tape motion end breakage of tape and also as a tape spool rotation counter. The device senses motion of the supply tape spool and delivers four pulses for each revolution of the spool.
The opto device is designed primarily for use with high speed tapes fitted with coded windows.
The main motor, which is electronically speed controlled, drives the capstan. As supplied, the deck operates at the standard tape speed of 1 7/8 in/sec (4.76 cm/sec) but with a slight modification to the circuit this can be made variable over the range 2.4 cm/sec to 9.5 cm/sec — approximately 15/16 in/sec to 3 3/4 in/sec. Wow and flutter are typically 1,15% r.m.s. (DIN 45507).
The deck also incorporates a unique, patented fast forward/rewind arrangement. To go from fast forward to fast reqind it is only necessary to reverse the current through the spooling motor. This arrangement is more reliable than conventional spooling drive systems as there are fewer mechanical moving parts. Spooling speed can also be varied from about 15 x to as much as 60 x normal forward speed simply by varying the voltage to the spooling motor. A stalled current limiter is incorporated in the spooling motor circuit.
For general purpose usage the machine is supplied with the B1202 two track mono record/playback head and the C21ES18 erase head but the head plate has been designed to accommodate most of the specialised heads in the Monolith range.
The head plate and pinch wheel solenoid system is driven via a circuit arrangement which ensures that the heads are firmly in contact with the tape during record and playback. Pinch roller pressure is 300 ± 50 gms.
The C 1000 is 180 mm long x 145 mm wide x 60 mm tall, and weighs 1.4 Kg (3.1 lbs). Prices (inc. VAT) are:

C 1000 main unit	£ 42.49
Opto sensor	£ 13.80
Hall effect unit	£ 5.75

Monolith Electronics Co. Ltd.,
5/7 Church Street,
Crewkerne,
Somerset.
Telephone: (0460) 74321

(1969 M)



Attractive LEDs

The Mentor Leds are available in two sizes of 3 mm and 5 mm diameter, with the option of four colours. Either size is offered with the choice of concave or convex reflectors finished in chrome or black.

In addition to the standard range described above, West Hyde are also able to offer many interesting variations. These include water resistant types, units fitted with wire leads and assemblies with built-in series resistors. Perhaps the most revolutionary addition to this LED range is the availability of the 'flashing' variants with all the circuitry built in. The most exciting types gives the ultimate function — steady green or flashing red from the one unit.

West Hyde Developments Ltd.,
Unit 9,
Park Streat Industrial Estate,
Aylesbury,
Bucks. HP20 1ET.
Telephone: Aylesbury (0296) 20441

(1967 M)

128 Character display module

This LCD 128 character display — model 1812, only requires an 8 bit microprocessor bus end strobe to control its upper and lower case character set. It uses a 7 x 5 dot matrix format end an independent controlled cursor line.
Ideal for use with all microprocessor systems, this CMOS module provides a truly cost effective solution and features a 2 page memory buffer and a cursor control as well as a back light for character illumination. It consumes little power and can be run on a single voltage supply (+5 V).

Also available es 80, 64, 40, 32 and 16 character models.

FELTEC,
Queensway,
Stem Lane Industrial Estate,
New Milton,
Hampshire.
BH25 5NN

(1960 M)

LCD hand-held multimeter

The TM354 from Thandar is a very compact 3½ digit, hand-held, pocket multimeter featuring a large ½" liquid crystal display, 0,75% basic accuracy, and 2000 hours battery life from one 9 V Alkaline battery.
The meter provides five functions in 14 ranges permitting measurement of DC volts (from 1 mV-1000 V) AC volts (from 1 V-500 V) DC current (from 1 μA-2000 mA) resistance (from 1 Ω-2 MΩ) plus diode check.

Supplied with test prods and protective vinyl pouch, the TM354 is designed end manufactured in England, and carries a full 1 year warranty. The TM354 cost £39.95 plus VAT.

Sinclair Electronics Ltd,
London Road,
St. Ives,
Huntingdon.
Cambs PE17 4HJ,
Telephone: (0480) 64646.

(1955 M)

Space misers

A range of space-saving storage cabinets and cases is available from Edward Roland. Designed for a multitude of uses, these very attractive storage cases provide tidy storage for all hobby accessories. Their lightweight, durable, all-plastic construction make them useful for keeping everything ship-shape in the garage, office, or workshop and there's no tidier way of storing things around the house.

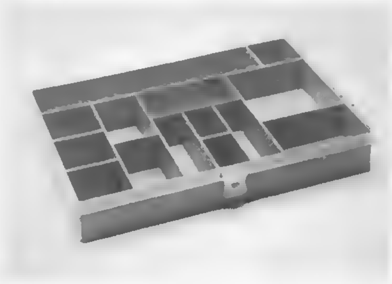
Multi Drawer Storage Cabinat

With practicality in mind, these cabinets are designed so that they can be stacked 2 or more on top of each other without slipping and also have the facility for fixing to the wall. Attractive see-thru drawers for fast content identification can be left in a partly or fully extended position without support, or removed for easy access.

Approximate dimensions: 11¾" x 5½" x 5¾"
Retail Price: £7.75

14 Compartment Case

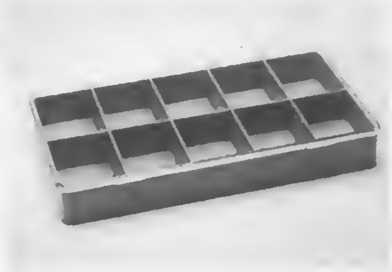
Beige in colour, this spacious case has a crystal-clear hinged lid with strong snap shut clasp and 14 various-sized compartments.



Approximate dimensions: 13¾" x 9½" x 2"
Retail Price: £5.75

10 Compartment Case

Also beige in colour, this tidy case has 10 equal sized compartments.



Approximate dimensions: 10½" x 5½" x 1¼"
Retail Price: £2.40
All items are aveileble by post and prices include VAT and postage and packing.

Edward Roland Ltd.,
215 Putney Bridge Road,
London SW15 2NY.
Telephone: 01.870.4296

(1972 M)

ELEKTOR BOOK SERVICE

sc/mputer (1)
build your own microprocessor system
formant
music synthesiser
sc/mputer (2)
build your own microcomputer system
ELEKTOR BOOK 75
300 circuits
digibook
a practical introduction to digital electronics
the practical introduction to a powerful system
Elektor
Elektor Publishers Ltd

JUNIOR COMPUTER BOOK 1 — for anyone wishing to become familiar with (micro)computers, this book gives the opportunity to build and program a personal computer at a very reasonable cost.
Price — UK . . . . £4.25 Overseas £4.45

JUNIOR COMPUTER BOOK 2 — follows in a logical continuation of Book1, and contains a detailed appraisal of the software. Three major programming tools, the monitor, an assembler and an editor, are discussed together with practical proposals for input output and peripherals.
Price — UK . . . . £4.75 Overseas £4.95

300 CIRCUITS for the home constructor — 300 projects ranging from the basic to the very sophisticated.
Price — UK . . . . £3.50 Overseas £3.70

DIGIBOOK — provides a simple step-by-step introduction to the basic theory and application of digital electronics and gives clear explanations of the fundamentals of digital circuitry, backed up by experiments designed to reinforce this newly acquired knowledge. Supplied with an experimenter's PCB.
Price — UK . . . . £5.00 Overseas £5.20

FORMANT — complete constructional details of the Elektor Formant Synthesiser — comes with a FREE cassette of sounds that the Formant is capable of producing together with advice on how to achieve them.
Price — UK . . . . £4.50 Overseas £4.70

SC/MPUTER (1) — describes how to build and operate your own microprocessor system — the first book of a series — further books will show how the system may be extended to meet various requirements.
Price — UK . . . . £3.70 Overseas £3.90

SC/MPUTER (2) — the second book in the series. An updated version of the monitor program (Elbug II) is introduced together with a number of expansion possibilities. By adding the Elekterminal to the system described in Book 1 the microcomputer becomes even more versatile.
Price details on order card.

BOOK 75 — a selection of some of the most interesting and popular construction projects that were originally published in Elektor issues 1 to 8.
Price — UK . . . . £3.50 Overseas £3.70

When ordering please use the Elektor Readers' Order Card in this issue (the above prices include p. & p.)