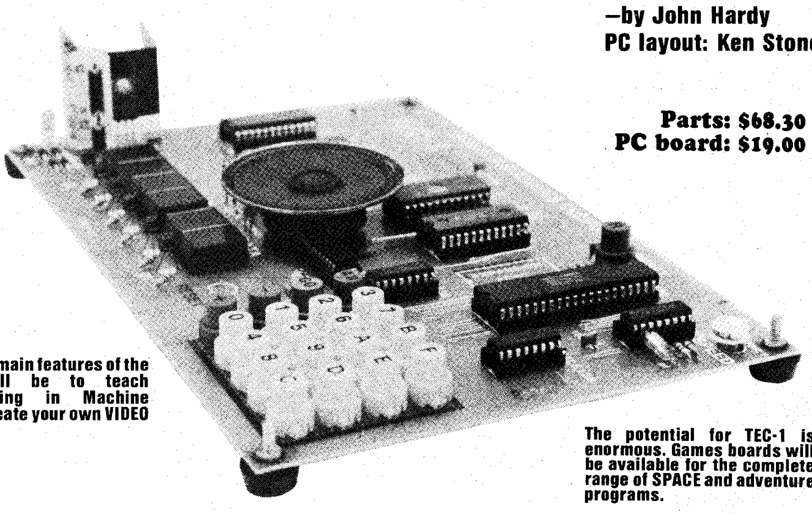
TEC-1

TALKING ELECTRONICS COMPUTER

—by John Hardy
PC layout: Ken Stone.

Parts: $68.30
PC board: $19.00



One of the main features of the TEC-1 will be to teach programming in Machine Code to create your own VIDEO GAMES.

The potential for TEC-1 is enormous. Games boards will be available for the complete range of SPACE and adventure programs.

If you think TALKING ELECTRONICS Magazine is a good place to start learning about electronics, you will find our TEC-1 computer absolutely fantastic.

We have spent many hours looking into the type of computers on the market and also computer kits.

Nothing has come up to the capabilities of the unit we are about to describe. And more important, you will learn the facts and operations of programming from ground level. We will assume you know nothing and thus place special attention to covering the meaning of every term and feature as it comes up.

The only requests we make are the following:

You must have already constructed at least 6 projects from Talking Electronics or equivalent magazines and it would be nice for you to have built the DIGI CHASER and say a couple of equally difficult projects such as the LOTTO SELECTOR and CLOCK.

This means you will be accustomed to soldering fine connections and know how to prevent making bridges between lands.

Fortunately the computer board has a solder resist mask and this means only the individual solder lands are exposed and they are already pretinned for easy soldering.

However some of the lands are close to one-another and a small low-wattage soldering iron is required for the project.

We have built 4 final designs and they all work perfectly. On one board we accidently created a solder bridge and this needed a little trouble-shooting, but we finally found it. So, for this reason, each kit includes a length of de-solder wick to mop up the surplus solder.

If you don't have a small soldering iron, fine solder and desolder wick, they will have to be obtained before constructing the kit.

BUYING THE KIT

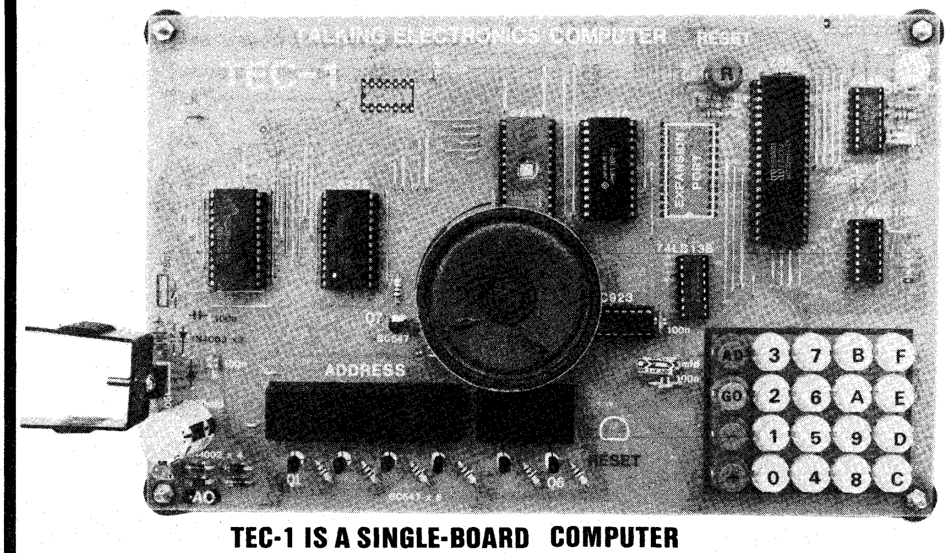
One thing you may not be aware of, is the need for one special chip.

Every computer requires a specially programmed chip so that it will start up and execute the correct operations. This chip can be likened to the BOSS in a work establishment. The chip we are referring to is the 2716 EPROM. You can buy it quite cheaply at any electronics store but unfortunately it is BLANK. And obviously it won't do a thing if you put it into a computer. To be of any use you will have to program it or write a program for it yourself.

Obviously this is way out of the question and so you have to buy one which is pre-programmed, from us.

For this service you have to pay a programmer's fee. A lot of time has

cont. P 60....



TEC-1 IS A SINGLE-BOARD COMPUTER

AN OVERVIEW:

The TEC-1 is a single-board computer with readouts in the form of 7-segment displays. The complete unit is shown in the photograph. It contains its own on-board regulated power supply which needs only an AC input for the computer to be fully operational.

The key pad is constructed from individual switches inscribed with hexadecimal numbers 0 to F and 4 switches labelled AD for address, GO, + for incrementing the address and − for decrementing the address.

The computer will play a number of games as well as present the alphabet and all this is contained in the 2716 EPROM which is directly above the speaker. The TEC-1 can also be connected to 8 output devices and they can be turned on and off in any combination as determined by the program you write. This program is stored in the 6116 RAM and any information in this chip is lost when the computer is turned off.

The reset button above the empty expansion port socket will reset the computer to the first address location (0800) and by pushing the GO button TWICE, any program you have entered into the computer, will run.

The computer contains 2k of RAM and this is programmed in machine code. Machine Code is very memory efficient and has a fast execution rate, making it possible to create high-speed programmes for video games and multi-function controlling.

Extra memory can be added via the expansion port and this is added to a daughter board directly above the main board via a dip header plugging into the expansion port socket. This will increase the capabilities of the computer to 12k plus 2k of memory-mapped in/out ports.

The speaker has two functions. It gives an audible beep every time a key is is pressed and becomes the output when music or tones are being played.

All the names of the chips are written on the overlay of the board and in simple terms they provide the following functions:

8212 - drives each digit for the display via buffer transistors.
8212 - drives the segments A - G and the decimal points for the display.
2716 - EPROM (Erasable Programmable Read-Only Memory). This has been programmed by John Hardy and contains the brains of the TEC-1.
6116 - The RAM (Random Access Memory) into which you put your own program. The Z80 also uses it during the operation of some of the programs.
Z80 - The heart of the computer.
4049 - The oscillator or CLOCK for the TEC-1.
74LS138 - selects between ROM (2716) and RAM (6116).
74LS138 - Selects between keyboard and display.
The photograph has been illuminated from the rear to show the tracks on the underside of the board. Normally these tracks are hardly visible as they are hidden under the solder mask.
Notice how neat everything is presented. You can credit the superb layout to Ken Stone who recognises the importance of making a project look appealing. Note especially the few resistors and capacitors required for a fully digital project.

The 20k cermet pot has been specially chosen as it has a cover which is connected to the wiper contact so that the pot can be turned with your fingers. This controls the speed of the operation of the computer and you will be using this control quite a lot.

The output pitch of the notes will vary according to the setting of the speed control as will the difficulty of the games and the scrolling of the screen when the letter sequence is addressed.

All chips are mounted in sockets for a mumber of reasons:
1. It looks professional.
2. It makes construction easy,
3. It makes testing and replacement easy, and
4. You can test other chips in the sockets.

The 100n capacitors are miniature solid dielectric types, about the size of a match-head, and they are specially suited to removing any spikes generated by the chips or from the power supply.

The TEC-1 will operate from a 6v battery such as a 509 lantern battery or from the mains via a transformer. The 7805 regulator keeps the operating voltage at 5v which is absolutely necessary for the chips we are using.

The battery back-up arrangement means you can have a battery sitting beside the computer in case the power fails or if you wish to change the computer from one room to another. When the battery back-up is operating the complete TEC-1 is operating as it is not posssible to power-down the Z80 without it affecting the contents of the RAM.

There are two empty IC sockets as well as a number of rows of holes on the board. These are for later expansion and not used at this stage.

The RESET key can be positioned near the display is desired. It is connected via 2 jumper leads to this lower position.

Finally you will be pleased to know the TEC-1 doesn't need any TV monitors, additional keyboards or bulky power supplies. It is self-contained on the single PC board.

THE EXPANSION PORT

The expansion port socket can be used in two different ways.

1. It can be used to increase the on-board memory of the computer to 4k RAM by inserting a 6116 RAM with IC socket, directly into the vacant space.

2. Alternatively, the expansion port can be used to increase the memory on steps of 2k by adding a daughter board above the main computer board. This will take a row of 5, 6116 chips and a bank of latches. Each 6116 will provide 2k of RAM and this is one of the add-ons which will be described in the next issue.

Each of the chips on the daughter board is selected by a line from the 74LS138 (near the clock oscillator). It is known as an address decoder and the first decoded output selects the EPROM. The second output selects the on-board 6116, the third selects the expansion port socket. If a duaghter board is used, the first chip on the board is selected and so on until 7 lines are used. 5 individual wires must be taken to the daughter board to provide this selection feature. They are taken from the 5 unused holes near the 74LS138.

To give an indication of the amount of memory you may require, here is a simple guide:

Each 6116 will accept 2048 bytes of information. A normal program contains between 1 and 4 bytes of data per instruction and this means one 6116 will accept about 600 instructions! To hand-assemble a program of this length wuld take months. We have only 3/4 filled the 2716 and you will be amazed at the capabilities of its contents.

So you can see, 2k will be quite adequate for most purposes.

The main use for the expansion is when the microcomputer is colecting and storing its own data for later retrieval. In this mode the computer can use up an enormous amount of memory, very quickly.

Take an example of a music sequencer. 2k of memory will last about 10 to 20 seconds. Or an echo unit. This will last less than 1 second!

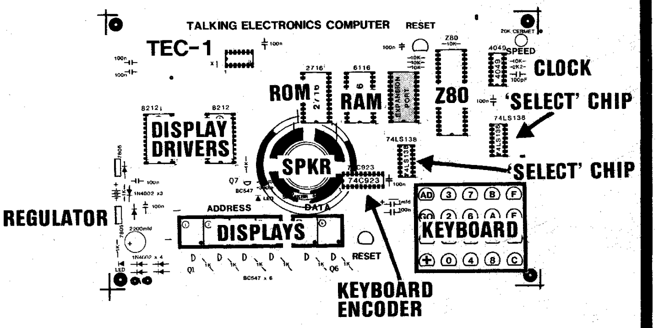
BATTERY BACK-UP

To use battery back-up, diode A must be installed. Connect the battery via a switch so that you can move from one location to another. Switch the battery OFF when the computer is using the mains power.

MARKING THE KEY-TOPS

The key tops can be lettered using LETTRASET. 16pt letters and numbers are used for 0 - F and 12pt letters for the AD and GO keys.
A coat of nail varnish will stop the lettering from wearing away.

THE FUNCTION OF EACH CHIP



TEC-1 is a complete microcomputer on a single PC board.

The function of each chip will become clearer after reading the text.

The most important concept is to understand how each chip is controlled by the Z80.

The above diagram shows where the ROM, RAM, Z80 etc are postioned on the board along with the other chips and devices.

been spent to get a set of instructions into the 2716 EPROM and each is individually filled from a master and verified, before it is added to a kit. This takes time and royalties are due to the designer, like the sale of a book or record. This accounts for its high cost.

We have called the EPROM a 2716 but actually it is a 2716-MON-1 EPROM, indicating it contains a program.

This is the only expensive chip. All the others have been chosen for their low price and availability. This is the way we approached the design of the computer. We looked at the price of each component and arranged the design around the low priced items.

The only components special to the TEC-1 are the EPROM and the printed circuit board. All the other components can be purchased at major electronics shops. The only advantage with buying a kit is the saving in time and frustration.

It would be very rare indeed for you to be able to buy all the components at one electronics shop. And so you will have to spend time and money in the hope of saving money.

In addition, there are a couple of pitfalls for the inexperienced. For instance, 4049 chips made by Fairchild should be avoided. They do not work in our situation. Also the push buttons should be the type suggested as one of the links inside the switch is used to create the wiring for the matrix.

The price structure for each kit is broken up as follows:

1. The Printed Circuit Board.
2. The 2716 MON-1 EPROM.
3. The kit of components.

The only other parts you will need to purchase are a 2155 transformer and power lead or a 9v DC plug pack rated at 500mA. You can, of course, use a lantern battery. This will be sufficient to operate the computer for about 5 to 8 hours, but will prove to be a very expensive way of running the TEC-1.

WHAT THE COMPUTER WILL DO

When you are going to spend a lot of money on a project and a considerable number of hours in its assembly, it's nice to know what the project will do.

Here is a summary of the first stage of the capabilities of the computer. Apart from the obvious experience gained in assembling a computer, the TEC-1 will make you aware of the chip-types required to create a complete system.

You execute some simple programs which use pre-programmed information from the EPROM and display it on the screen. If we take the letter program for instance, you create a single static letter,then add another letter and enable them to run across the display. Finally you create your own words and sentences which can be made to pass across the display at a rate determined by the setting of the SPEED control.

But most important you learn some of the instructions necessary to write your own programmes.

You also learn to increment and decrement the memory address to look at the contents of each location and possibly alter it if required.

You carry out the same procedure with a set of tones and these can be combined to produce a tune. You can also access two tunes in the ROM and this will give you an indication of what can be achieved. Your own tune can be added to the end of the RUNNING WORD DISPLAY and create a wide variety of possibilities.

There are also three games in the EPROM and these can be played in-between the educational programming.

The first game is NIM. Everyone knows this game as 23 matches. The address location for this game is 03E0 and the computer starts with 23 on the display. The object of the game is to try and leave the computer with last match. You can take 1, 2 or 3 matches during your turn. Believe me, it isn't easy.

The second game is LUNA LANDER. Its address is 0490. The numbers on the screen represent velocity and height. You are required to land on the surface of the moon at zero velocity without running out of fuel. The full details of this game are on the last page of this article.

The third game is INVADERS. A number is set up on the left hand end of the display and invaders approach from the right. See the article on how to exterminate them on P 74.
Difficulty is set by the speed control and your score apprears at the end of your turn.

This is only the beginning. In the next issue we will expand the TEC-1 and interface it with the outside world.

PRICES:

Here are the prices for the TEC-1.

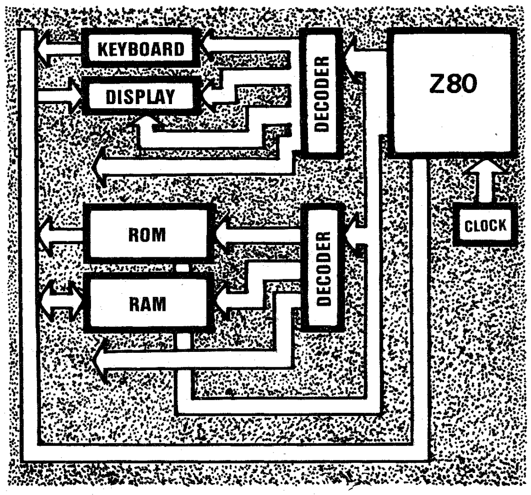
Depending on how much you already have in stock and how you intend to construct the project, so the price will vary. Don't spoil the ship for a ha'penneth o' tar. Use only the best components.

Complete kits are available at our larger outlets and the PC board will be available with pre-programmed EPROM from some of the other outlets. As a back-up service, the complete kit will also be available through the magazine as listed on the kit pages. Remember, the board is double screened and solder masked to give a classy finish to the project. If you have never made your own PC boards before, don't start with this board. The work involved in making a board of this complexity is enormous and the result will be nothing like the cover photo.

The only two outlays you have to accept are the PC board and the pre-programmed EPROM. All the rest can be obtained from your local supplier.

PC Board $19.00 (post $2.50)
2716 - MON - 1 Programmed EPROM **$12.00**(Post $1.50)
Kit of Parts (including EPROM)**$68.30** (Post $2.50
All Parts & PC Board $87.30 (Post $4.50)
You will also need a 6v lantern battery (from your local hardware shop) or a 2155 transformer ($5.90) and a power lead or a 9v AC or DC Plug Pack rated at 500mA ($14.50)

Complete TEC-1 with transformer $93.20 (Post $5.50)
Complete TEC-1 with Plug Pack $101.80 (Post 5.50)



TEC-1 BLOCK DIAGRAM

This simplified BLOCK DIAGRAM shows how each of the chips are inter-connected.

The Z80 Central Processing Unit is the overseer of the whole system and it selects which device it wishes to access via one of the 74LS138 decoder chips. Each will select one-of-eight output lines. These decoder chips are not fully utilized in this project and this leaves room for further expansion.

If we take the key-board as an example, we see it passes its information to the Z80 via the DATA BUS. This bus consists of 8 lines and carries binary information. This will allow any number from zero to 255 to be sent.

The ADDRESS BUS is a 16 line path which is only a one-way street. Information only emerges from the Z80 on this bus. The Data bus is a two-way street of 8 lines. Information can be passed into the Z80 on this path as well as emerge from it.

Each block in the diagram represents a chip and the only two chips missing are the display drivers.

OUR BORDER

Z80 computer terms have been added to the top and bottom of the pages, for this project. Some of the more common instructions are contained in this string.

Here are the meanings of these terms:

ADD Add the contents of a CPU register to the accumulator.

AND This is a logical AND operation in which two binary numbers are compared. If the first digit in each number is a 1, the answer is a 1. If only one number is 1, the answer is 0. If both numbers are 0, the answer is zero.

BIT This is an instruction to test the status of a bit in a register.

CALL This is a call instruction which will be executed if a particular condition is satisfied. If the condition is not satisfied, then the call instruction is ignored and the program execution continues.

DEC This is an instruction to decrement the value of a CPU register by one.

EX This is an exchange command. The contents of any register can be exchanged with any others.

IN Input to a CPU register from an input port.

INC The increments the value of a CPU register by one.

JP This is a jump instruction.

LD This is a load instruction.

NEG This instruction negates the contents of the accumulator. The result is the same as subtracting the contents from zero.

NOP This is the NO OPERATION instruction.

OR This is a logic instruction which compares two numbers. If either of the first digits is a 1, the answer is a 1. The same applies to the second and third digits. etc.

OUT An output instruction from a specified CPU register.

POP This is an instruction to POP from the stack into a register pair.

PUSH This is an instruction to PUSH an index register onto the stack.

RES This is an instruction to clear the status of a single bit in a CPU register to the logic zero state.

RET This is a conditional return instruction.

RL This rotates the contents of a CPU register to the left through the carry bit.

RST A restart subroutine directive

SBC A double subtraction instruction

SET Sets the status of a single bit in a CPU register to the logic ONE state.

SLA This instruction shifts the contents of a memory location to the left

SRL This shifts the contents of a CPU register to the right

SUB This instruction subtracts the contents of a CPU register from the accumulator

XOR This is a logic instruction which compares each bit of two numbers and gives an answer of 1 if either bit is one. But if both are 1, the answer is zero.

LOOKING INTO THE TEC-1

There are two chips in our computer which provide the major amount of processing.

To make it easy to understand, we will call them the BOSS and WORKER. The boss is the 2716 which is the specially programmed Read Only Memory (ROM) and contains all the information to get the computer started and keep it operating.

The worker is the Z80. It is the arms and legs to which all instructions are sent and it provides the ability (muscles) to carry out the requests of the ROM.

The Z80 can also be thought of as an octopus, extending out its tentacles to all parts of the computer to keep everything in very strict control.

These two chips are the most important items in the computer and it will almost run without any other devices. But you would not be able to push any buttons or see the results of the operations. So we need more chips.

The first of these are the display chips. Because each has only 8 outputs, we need two. The display is multiplexed (see issue 2. P.5.) and this type of design uses the least amount of wiring and the least number of input leads. For a 6 digit display with decimal points, we require 8 inputs for the segment drive and 6 inputs for the digit drive. One 8212 is used for each of these with the digits being driven via driver transistors.

This leaves two spare outputs and one of these is used to drive the speaker via a buffer transistor (Q7).

The Z80 (the microprocessor) is constantly feeding information into the display via the pair of 8212's. When you understand the operation of multiplexing a display you know it is constantly being scanned to create the figures.

To prove this feature, turn the speed control down to minimum and shake the board. You will be able to detect the strobing of the display.

The keyboard is also an interesting feature. It is also being constantly scanned by the 74c923 (the chip near the speaker), waiting for one of the keys to be pressed.

The scanning commences at the first row, which is the bottom row and it reads from each of the columns to see if any of the buttons have been pressed. Next it progresses to the second bottom row and again checks the columns. If a button is detected, it sends a debounced signal to the Z80 and interrupts it. The Z80 drops whatever it is doing and accepts a 5 bit binary number from the 74c923, which corresponds to the key being pressed. An example of a 5-bit binary number is 10011 and the following table gives the value for each key on the pad.

KEY	Binary No.
0	00000
1	00001
2	00010
3	00011
4	00100
5	00101
6	00110
7	00111
8	01000
9	01001
A	01010
B	01011
C	01100
D	01101
E	01110
F	01111
+	10000
-	10001
GO	10010
AD	10011

The 2716 ROM tells the Z80 how to interpret the 5-bit binary instruction and what to do with it.

This means we could change the position of all the keys, re-program, the 2716 and the system will be operational again. In other words it is a SOFTWARE programmed set of instructions.

The 74LS138 below the EXPANSION PORT selects between the keyboard and display. Take this example: Button 5 is pressed. The Z80 has all its attention directed to scanning the display via the pair of 8212's. The 74LS138 is allowing the 8212's to function and at the same time prevents an output from the 74c923 to be passed to the Z80.

When button 5 is pressed, the 74c923 sends and interupt signal to the Z80. The Z80 stops scanning the display, requests the 74LS138 to shut down the display and open up the information from the 74c923 from the keyboard. Once the keyboard is read, the Z80 reverts to scanning the display.

This happens so fast that you cannot see the display flicker. The blanking you may see on your model is the result of the time taken to beep the speaker. The longer the beep, the longer the displays are blanked.

The RAM is like a black board. It holds temporary information which is being constantly modified by the microprocessor. When the power is switched off the contents of the 6116 is lost.

The 2716 has a set of instructions which allows the user to access the RAM. Without these instructions you would never be able to get into its memory.

The 74LS138 near the speed control selects between the ROM and the RAM and also any extra memory added to the expansion port.

The 4049 is simply an oscillator or clock which produces clock pulses for the Z80.

Because the Z80 is a dynamic device, its registers need to be constantly cycled to retain their contents. There is a minimum clock rate for this and if the rate is reduced, the computer will crash.

SUMMARY OF EACH CHIP

8212 - Display driver. One chip supplies 8 lines to the segments and the other drives the digits via a driver transistor.

2716 - ROM. The central library of the computer. It tells the computer how to operate.

74c923 - Keyboard control device which interfaces the keyboard with the Z80 chip.

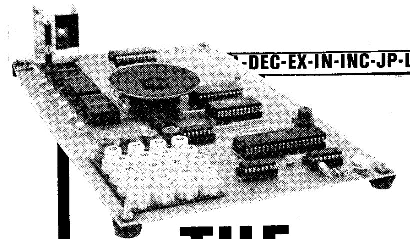
6116 - RAM. Temporary storage for data and instructions. Storage for your own programmes.

74LS138 - Selects the display or keyboard as required by the Z80 chip.

Z80 - The core of the computer. Contains all the necessary logic for executing programmes and manipulating numeric data.

4049 - Wired as an oscillator and fed to the Z80 to determine the operating speed of the system.

74LS138 - Selects between the ROM and RAM as instructed by the Z80.



THE Z80 CPU

The heart of the TEC-1 is a Z80 CPU. This is the largest chip on the board, having 40 pins, and is the central item around which all the other chips operate. The 40 pins are all used to advantage as they are needed to send and receive data as well as send out address locations. On top of this, 8 pins are required for controlling the functions of the Z80.

If we consider the Z80 to be a worker, the BOSS will have to be the 2716 EPROM.

When the computer is first turned on, the Z80 has just enough intelligence to output an address to the EPROM to locate the CPU's first instruction. The EPROM returns this instruction via the DATA BUS and the two start communicating. The EPROM tells the Z80 what to do, how to do it and where it must be put. In less than a second, the start-up procedure has been completed and the whole system comse alive with the START ADDRESS appearing on the display.

For the moment, the Z80 is the chip we wish to investigate.

Most of its pins are ADDRESS and DATA lines. Eight of these are grouped together to become the DATA BUS and 16 are grouped together to become the ADDRESS BUS.

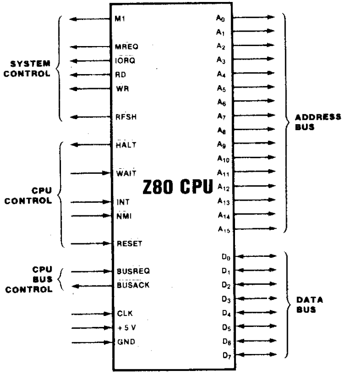
The Z80 uses the ADDRESS BUS to locate the data it wants. This may be in the ROM (the 2716) or in the RAM (6116). It uses the ROM/RAM select chip 74LS138 in this process. Or the information may be from the keyboard. In this case it uses the display/keyboard select chip, another 74LS138.

The Z80 can only do one thing at a time and it is only because the system is operating at between 250kHz and 2MHz (as determined by the speed control) that you think everything is happening at once.

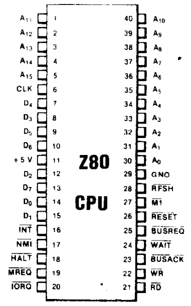
THE Z80 SERIES

The Z80 series (without an 'A' after the 80) was the first to be developed and has a maximum operating speed of 2.5MHz. The Z80A series operates at 4MHz.

There is a whole family of Z80 chips and you must be careful to read the letters which follow the Z80 name, to identify the actual function of the chip.



Z80 LOGIC FUNCTIONS



Z80 PIN OUTS

The Z80 microprocessor is the central element of a microprocessor (computer) and is called CPU. This stands for Central Processing Unit.

Five other chips provide support for the CPU in complex computer systems. We have not used any of these in our simple system but it is handy to know of their existence.

They are: The PIO (Parallel Input/Output). This can be wired to interface peripheral devices such as printers and extra keyboards etc.

The CTC (Counter/Timer Circuit). This chip features 4 programmable 8-bit counter/timers each of which has an 8-bit prescaler. Each channel will operate in either counter or timer mode.

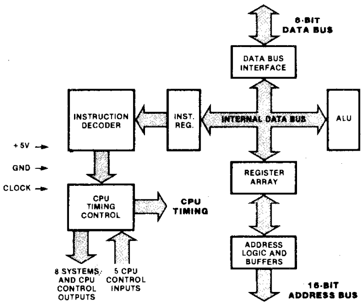
The DMA (Direct Memory Access). This controller provides dual port data operations.

The SIO (Serial Input/Output). This controller offers two channels. It is capable of operating in a variety of programmable modes for both synchronous and asynchronous communication.

The DART (Dual Asynchronous Receiver/Transmitter). This chip provides low-cost asynchronous serial communication. It has two channels and a full modem control interface.

The Z80 series of chips have completely different operations from each other and the letters on the chip are VERY important.

It's difficult to realize, but the Z80 is classified as a "dumb worker". It may be dumb but it is very quick. It is capable of carrying out instructions at the rate of about 10,000 to 200,000 operations per second, depending on the type of instruction. Each of these takes a particular number of cycles to execute and these features are contained inside the Z80 architecture and cannot be altered.



Z80 CPU BLOCK OIAGRAM

Each operation for the Z80 has a machine code instruction such as 1E, 06, 0E, 85, E6 dd, 06, E9, 8, 81, ED 47, 00 etc and these will be discussed in a later article. For the moment, we want the computer to seem a reality.

There are lots and lots of sections inside the Z80 chip and most of them are very difficult to explain in simple terms.

One area which can be explained via a simple comparison is the bank of registers. These perform most of the operations in the Z80.

They are given the names B, C, D, E, H and L, with an accumulator register A. These registers can be likened to a car space in a parking lot. Each register represents one car space. The car represents one WORD of information and this work consists of 8 bits or one BYTE. A bit is a signal on a single line which can be either HIGH or LOW and 8 lines enter the Z80 in the form of a DATA BUS.

This data bus is the same as the road into a parking lot and the car is one word. The 8 bits are 8 seats and each car can have up to 8 people. Depending on where they sit and the number of people, the size of the byte is determined. We can say that byte and word are the same for our system as the Z80 is an 8-bit microprocessor. If it were a 16-bit microprocessor, a word would be 16 bits.

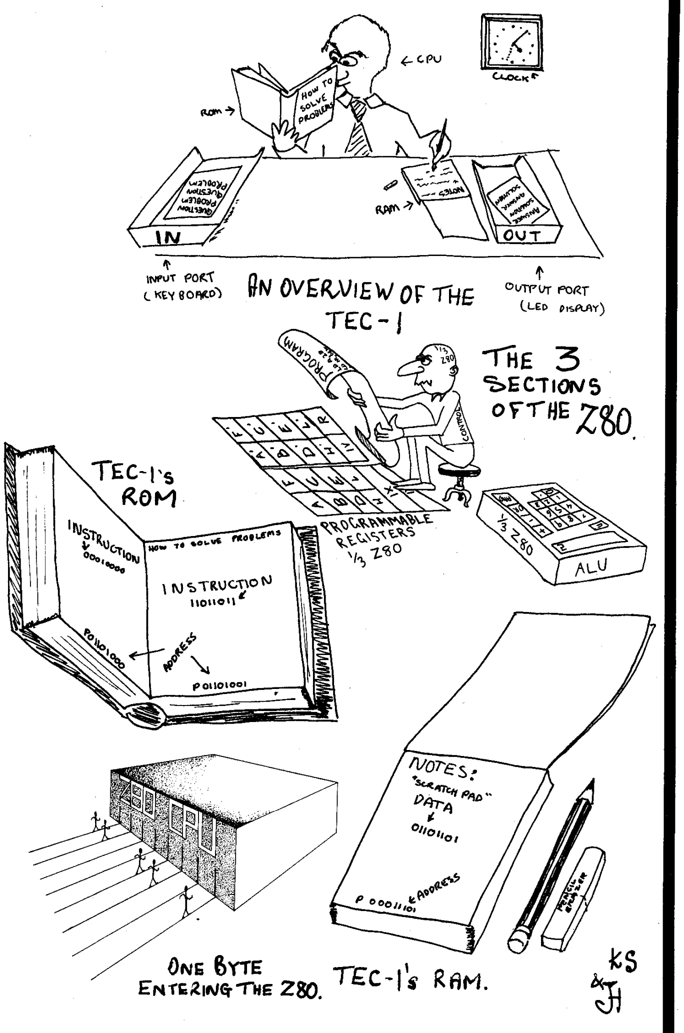
Normally the car parks in space A (register A) and this is also called the accumulator register as the answer for any addition instruction, for instance, will appear in register A.

The car can represent a number from 00 to 255 and the register will accept any of these numbers. This is all the register will hold. . .just one number from zero to 255.

There is one important fact that we have omitted to mention. Before the byte can be put into register A we must send an instruction to the Z80 so that it will know where the number is to be put.

This instruction happens to be 3E for register A. If you wanted to load register B, the operation code would be 06 and to load register C it would be 0E. These codes are called "MACHINE CODES" or MACHINE CODE LANGUAGE and they are interpreted by the Z80 to perform one of over 245 different shuffling or arithmetic operations.

If you want to add a number to the number above, it will have to be firstly loaded into register B, then the two registers can be added. This will take a number of operations with the result always appearing in register A.



You can transfer the contents of register A to the RAM (6116) via a further instruction so that the result is not lost when the next number is sent to register A. Otherwise the previous contents of register A are written over.

The Z80 contains an equivalent bank of emergency registers which can be accessed via a special instruction. These are called A'(A-prime) B', C', D', E', F', H', and L'. It also contains a number of 16-bit registers (like a space for car and trailer) and numerous building blocks which are needed to keep the Z80 operating. These can be likened to the workers needed to keep a parking lot neat and with a smooth flow of traffic.

CONSTRUCTING THE TEC-1

Constructing the TEC-1 is no more complex than building any of the cover-projects in Talking Electronics . . it only takes longer.

The most important aspect of this project is NEATNESS. We have gone to a lot of trouble to create a printed circuit board that looks really neat, with a layout that is very pleasing. Don't upset the aesthetics of the board with poor-quality layout or incorrect components. If you intend to buy the components individually at your local electronics store, look at the photos in this article for the type of components we have used, and purchase the same styles.

Don't use anything old or dirty and make sure the tinned copper wire for the jumpers is CLEAN, thick and absolutely straight. We will tell you how to do this in the notes.

The TEC-1 is not designed to be fitted into a case. It is too beautiful to hide. Like all our projects, it is designed to be viewed. This keeps you alert to the construction, contents and arrangement of the chips and components. You must constantly remind yourself of the name of each chip and its function. It's an indoctrination process which can only be of benefit in the long term.

Before commencing construction we suggest you get everything organised on the workbench.

We can't stress strongly enough, the need for a good soldering iron.

We have a range of soldering irons in our assembly area including: a 10watt, 12v pencil iron, a 15watt 240v Micron, a 60watt Constant Temperature Scope iron and a Weller Soldering Station. We also have a plummer's soldering iron and two instant-heat solderings as well as a miniature instant-heat iron. Which one would you choose?

If you chose an instant-heat iron, we don't want to know you. They will lift the lands off the board and create more problems than you can imagine. Also construction time will be considerably longer as you have to wait for them to heat up for every connection. Our choice is the 10watt 12v type. It it light-weight, and enables you to produce a speedy connection. This is important when constructing a large project like this.

Other important tools and aids are: fine solder, sharp side cutters, and a pair of long-nosed pliers. You will also need a soldering-iron stand and a solder tray to accept the dead solder left on the iron after making each joint.

Get everything ready on a clean part of the workbench and have all the components available for insertion.

The first part of construction is the most laborious. It is the fitting of the 55 links. You must take great care when fitting these links as they must be absolutely straight, with their ends bent to a sharp 90°. The whole link must touch the board.

Start at one end of the board. Cut a length of copper wire about 10cm long, which will be sufficient for about 5 links. With a pair of pliers at each end of the length of wire, pull the two pliers apart until the bends and kinks are removed and the wire is perfectly straight. Now you can use the wire. Bend one end with the pliers and insert it into one hole in the board. Solder this end and snip the excess wire from the joint.

Feed the other end down an appropriate hole and pull it through with the pliers until the link becomes straight. Keep this part pressed against the board while soldering it. Cut the surplus from the connnection and inspect the first addition to the board. Continue with each link as you come to it and take your time. It will take the best part of an hour and no link should be loose enough to touch any other, even if it is pushed slightly.

The next components to add to the board are the resistors. There are 15 of these and they should also touch the board. Check the value of each resistor before inserting it. They are hard to remove if you make a mistake.

Next are the IC sockets. The reason for inserting them at this stage will be quite obvious. As each socket is inserted, it becomes the highest component on the board and this means the board can be turned over and the socket will rest on the workbench while the pins are being soldered.

You almost cannot make a mistake with these sockets as the number of pins corresponds to the holes in the board. The only point to remember is the identification notch at one end of the socket. These should cover the dot so that when the chips are inserted, the notch on the chip aligns with the dot on the board.

Next add the 6 FND 500 displays. Once again the board can be turned over and rested on the display to keep it pressed against the display while the pins are being soldered.

The next order of insertion is not critical and would consist of inserting the power diodes, 2 LEDs, 7 100n

cont. P. 68

PARTS LIST

1	-	100R
1	-	330R
8	-	1k
1	-	2k2
5	-	10k
1	-	20k cermet
1	-	100pf
7	-	100n 100v
1	-	1mf 16v
1	-	2200mfd 25v
4	-	1N 4002 diodes
7	-	BC 547 transistors
1	-	5mm red LED
1	-	5mm green LED
6	-	FND 500 or 560 displays
1	-	7805 regulator
2	-	8212
1	-	2716 TEC-1 Monitor
1	-	6116
1	-	74c923
2	-	74LS138
1	-	Z80 CPU
1	-	4049 NOT Fairchild)
3	-	16 pin IC sockets
1	-	20 pin IC socket
4	-	24 pin IC sockets
1	-	40 pin IC socket
21	-	PC mount push switches
1	-	8R speaker
1	-	heat fin for 7805
4	-	rubber feet
5	-	nuts and bolts
60cm		tinned copper wire
3m		fine solder
10cm		desolder wick

Substitutes:
2200mfd electrolytic can be replaced by 1000mfd in the TEC-1.
Rubber feet can be stick on feet.

HOW THE CIRCUIT WORKS

The TEC-1 circuit looks very simple and, in fact, it is very simple.

This is because many of the chips in a computer circuit are connected to a parallel wiring system called a BUS. There are 2 main buses in a computer and they are called ADDRESS and DATA.

Normally the ADDRESS bus is 16 lines wide but our computer is only a baby design. We have used 11 lines in the address bus with line 12, 13 and 14 going to a decoder chip to select between display, keyboard, memory and expansion.

The data bus is like a highway with data passing to and from the Z80 and the other chips. The data line will carry a binary number between 00000000 and 11111111, which is 0 - 255.

The 8212's are latches which drive the displays. One controls the segments a to g and the decimal points while the other drives the digits via a set of buffer transistors.

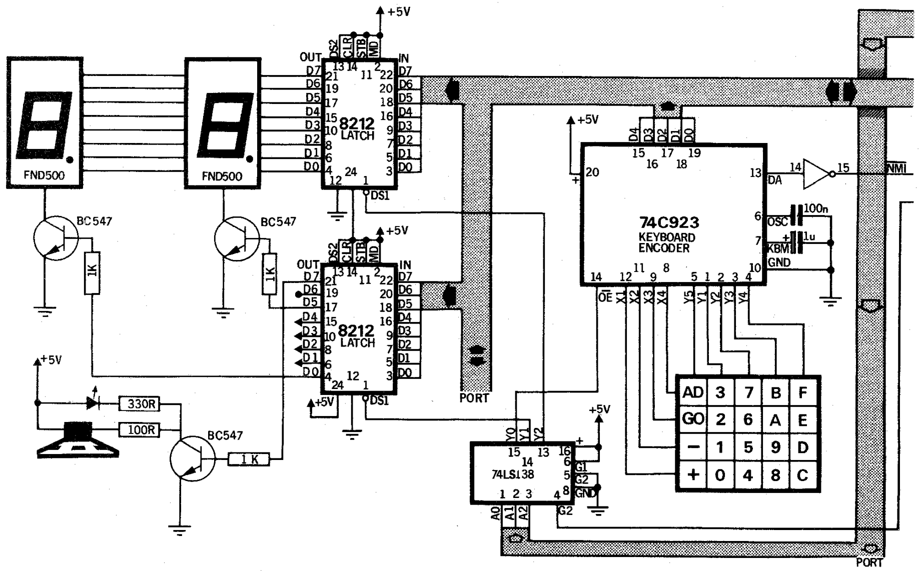
Data and program are stored in the memory whch comprises the 2716 EPROM and 6116 RAM.

The Z80 addresses a particular location in the memory by sending a binary number down the address bus. It determines the condition of SENDING or RECEIVING by the state of the R/W line (pin 22). When this line is LOW, the Z80 is sending data to the RAM and when HIGH, it is receiving data from the memory.

Data is sent or received via the data bus and only one data transfer can occur at a time.

When the reset button is pressed, the computer sets the condition for initial data entry. These include entering (or loading) the address pointer to 0800, setting the dots on the data displays, setting the stack to the highest point in the 6116 RAM and calling a routine to produce the two-tone 'ready' beep.

The Z80 then goes into a scan routine to display 0800. This information is

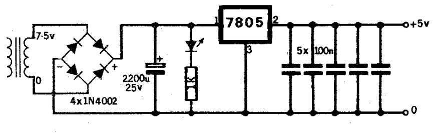


POWER SUPPLY

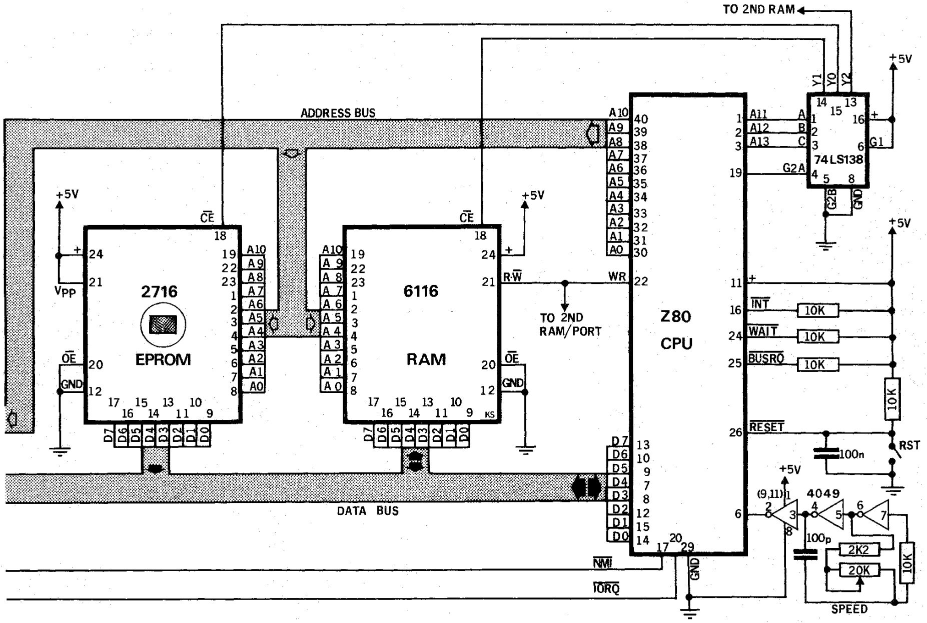
The 7805 regulator is included on the PC board. When using a 2155 transformer to power the TEC-1, use the 7.5v tapping. This will produce about 9.5v DC into the regulator which is ideal to gain full voltage and current from the power supply without overheating the regulator.

The TEC-1 will accommodate a DC Plug Pack. Use a 9v type capable of delivering 500mA.

The five 100n capacitors on the output line are spike suppression capacitors. They are placed near each of the chips to prevent noise from one chip upsetting the funtion of the computer. We suggest low-impedance mono-block types for this application.



POWER SUPPLY



taken from the highest bytes in the RAM, which have been deposited in the set-up routine.

The Z80 will continue to scan the displays until it is interrupted by the 74c923, via the inverter of the 4049 chip. This is a Non Maskable Interrupt line. The Z80 immediately branches to a routine at 0066H which inputsthe binary code of the key being pressed and stores it in a register in the Z80.

It firstly checks if the key is a function key or a numeric key 0 - F. It does this by checking bit '4' of the 5-bit binary number. Bit 4 is the 1 in: 10000. The first bit is called bit 'zero'.

If the display is in the address mode, the function key will simply put it into the data mode. This is indicated by the dots moving to the data displays.

If the computer is in the data mode, a function key will perform the function intended. A + will increment the address pointer , a — will decrease the address pointer AD will set is to the address mode and GO will execute the program starting at the address shown in the display.

If a numeric key is pressed, the data displays are cleared and the digit is shiftedin from the right. A second key will produce a 2-digit number.

When the + key is pressed when the displays are in the data mode, the address will increment.

While this seems a simple operation, an enormous amount of data is flowing from the Z80/ROM/RAM combination. The address pointer, which is temporarily stored in RAM, is loaded into the Z80 HL register (two 8-bit registers connected to become a 16 bit register). This register increments by instruction 23 (inc HL) and HL is then stored back into the same location in RAM.

The display is then changed to reflect its new address and contents of RAM (in data displays). The Z80 then reverts to its scan routine waiting for another key to be pressed. All this happens in a few milliseconds!

The 74LS138's are simple DEVICE SELECTING chips. The have 3 binary input lines and this enables them to select any one of 8 devices. These can be ROM, RAM, keyboard, display, speaker, or external chips, even additional memory or video displays.

The speaker is simply an amplified version of a 'BIT'. A BIT is a HIGH or LOW state and constant rapid changing from HIGH to LOW will produce a tone.

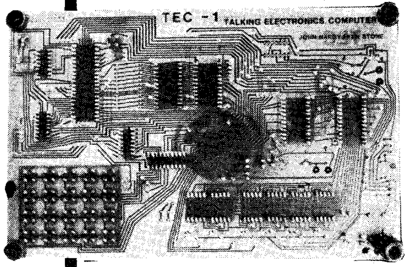
The quality of the sound can be altered by the ratio of the HIGH to the LOW. White noise is a result of random bit production. Correct programming enables the production of music and sound effects.

The power supply is a simple 7805 voltage regulator arrangement. Provided the input voltage is only about 3v above the output voltage, the regulator will not need a large heat-fin. The 2200mfd electrolytic can be replaced with a 1000mfd electrolytic as the computer consumes only about 500mA.

The speed of the TEC-1 is controlled by the 4049 clock oscillator. This is only a simple 2-inverter oscillator which can be adjusted via a speed control to vary the speed of the information passing the displays. A crystal controlled clock can be added at a later stage.

capacitors, 7 transistors, 1- 100pf capacitor, 20k cermet pot and 1 - 1mfd electrolytic.
Attach the flag heat-sink to the 7805 and insert the regulator into the holes nearest the 2200mfd electrolytic.

The other 7805 powers the expansion board and will be covered at a later stage. Insert the 2200mfd electrolytic and solder the leads.



The underside of the TEC-1 showing the layout of the copper tracks.

The keyboard switches are individually inserted and soldered as shown on the overlay. The flat on the switch runs across the bottom of the switch so that the jumper link inside the switch completes the wiring of the matrix.

Attach the speaker to the board via a piece of double-sided sticky tape and connect the voice coil to the circuit via short lengths of tinned copper wire.

Four rubber feet are attached to the board with nuts and bolts to prevent the underside of the board from scuffing the workbench.

The final, and most important items to add, are the IC's. These are pushed into the sockets so that pin 1 on each chip is facing towards the display. The 74c923 faces towards the left and you double check each chip before AND after it is inserted. If the rows of pins are too wide, they can be pressed closer by pressing the edge of the chip on the PC board and then the pins will be easier to insert into the socket.

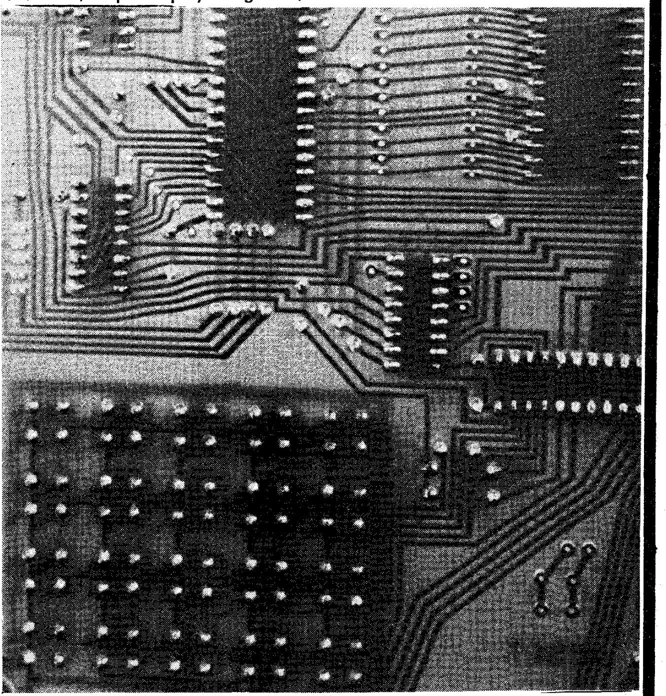
Connect an AC supply to the board and the TEC-1 is ready for operation.

You can use either a 2155 transformer or a 9v plug pack capable of delivering 500mA. In either case the incoming voltage should not be more than 8v to 9v to prevent the regulator getting too hot.

Switch on the power and note the display lights up with 0800. This is the first available address and indicates the computer is ready for action.

If you are well-versed in Machine Code language, you can begin immediately with preparing your own programmes. You will find the TEC-1 is very versatile in its applications and will allow a wide variety of expansions to be accommodated.

Treat the computer as a basis for experimenting and learning. Later we will provide add-ons for a crystal oscillator, output displays for games, and control devices for up to 8 different items at the same time.



An enlargement of the Key-board section showing the soldering.

We will also welcome any programmes you write for the computer and it doesn't matter what subject they are witten about.

The tape interface to be added in the next article will allow you to save programmes and re-use them later.

If you are new to programming, you will appreciate the introduction presented on P. 71. It starts at the beginning and shows you how to key a short program and activate a readout in the form of a visual display as well as a musical score.

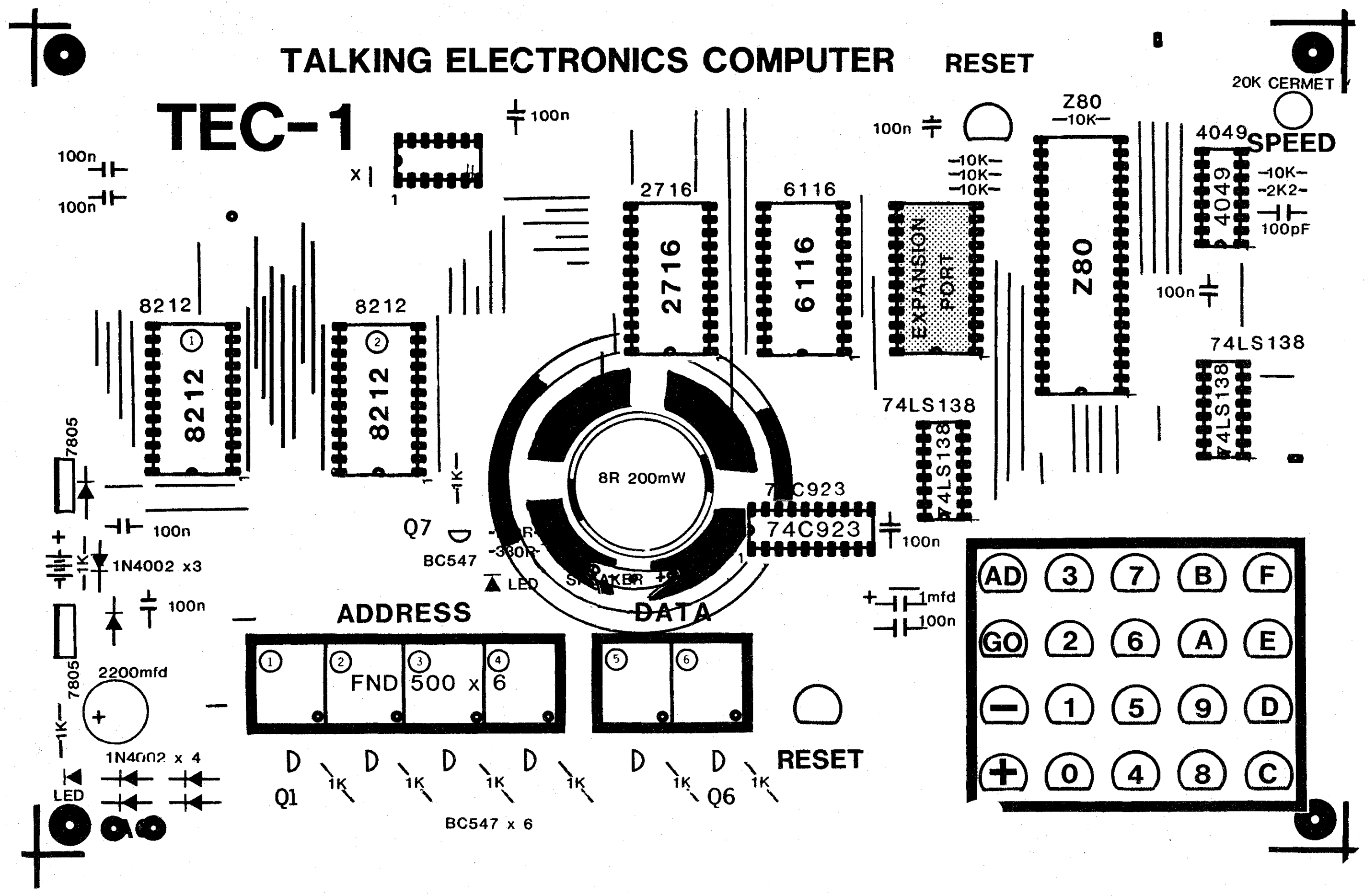
Three games on P. 74 will intrigue everyone. The level of skill can be adjusted by turning the speed control. This increases the rate of operation of the whole computer.

The are also other programmes in the EPROM and these will be discussed in the next article.

If you are having trouble getting the TEC-1 to operate, see the article on P. 70. It will solve most of the simple construction faults.

We all wish you the best with your new acquisition.

Don't under-estimate the capabilities of the TEC-1. It is a very powerful machine.


TALKING ELECTRONICS COMPUTER
RESET
TEC-1
100n
100n
100n
x1
1
8212
8212
① 8212
② 8212
2716
2716
6116
6116
EXPANSION PORT
100n
10K
10K
10K
Z80
10K
Z80
4049
4049
SPEED
20K CERMET
10K
2K2
100pF
100n
74LS138
74LS138
74LS138
74LS138
74C923
74C923
100n
8R 200mW
SPEAKER
330R
LED
Q7
BC547
1K
7805
7805
100n
1N4002 x3
1K
100n
2200mfd
1K
1N4002 x 4
LED
ADDRESS
DATA
① ② ③ ④ ⑤ ⑥
FND 500 x 6
D
1K
Q1
Q6
BC547 x 6
RESET
1mfd
100n
AD
3
7
B
F
GO
2
6
A
E
1
5
9
D
0
4
8
C

IF THE TEC-1 DOESN'T WORK:

If you are faced with the situation where the TEC-1 fails to operate properly, or if it doesn't work at all. . . don't worry. This will be a blessing in disguise.

You learn a lot more about electronics and computers by fixing the TEC-1, than just building and running it.

As requested in the introduction to this project, you should already have a certain amount of background in building projects. This is when all these skills will come together.

The first point to remember with the TEC-1 is this: The TEC-1 SHOULD operate perfectly the first time it is turned on. This is because it is built with NEW components which are first-quality items and the PC board has been thoroughly checked. If you are unfortunate enogh to produce a dud, you must firstly realise that there is a 99% possibility that the fault is in the construction.

You should go over the entire project again, checking every component, connection and the value of each part. The best way to do this is to ask someone ELSE to do the checking. This is because you cannot check your own work. Most of the projects that come to us for repair are simple faults, overlooked by the constructor. Faults like 1k instead of 1M, 22k instead of 3k3 etc. This type of fault can very easily creep in. This is because humans think positively. Most constructors are CERTAIN all the values are correct! How could they make a simple mistake like THAT?

After passing the TEC-1 over to a government checker (anyone impartial) you can begin the TEST procedure. This will need test equipment.

This is where the LOGIC PROBE will come in handy. That's why we presented it in this issue. You will find it invaluable, as most of the lines on a computer are PULSE lines and these are constantly changing accoding to the clock rate or as requested by the Z80.

The first test is a RESISTANCE TEST.

To carry this out successfully, you should remove all the chips. This is to prevent any false readings.

We will be looking for solder bridges between one or more of the pins. These can be very difficult to see as they are sometimes as fine as a human hair or even merely a microscopic splash of solder.

That's why you must never tap the soldering iron on or near the board, as excess solder will fly off the tip and land on some unknown part of the board. This will cause a bridge which will take hours to locate. You must only tap the iron in a solder tray and this must be done after every joint to prevent dropping solder and creating a problem, like now.

Set the multimeter to LOW OHMS RANGE. Make sure you adjust the ohms control so that the needle travels to the far right hand end of the scale to indicate very LOW resistances.

When all the chips are removed, most of the wiring on the underside of the board consists of individual conductors and this means almost no adjoining pins are connected. This makes it ideal for testing via a resistance measurement.

The first place to check is the RAM/ROM section where each of the pins has a conductor running between them. Turn the board over and measure the resistance between each solder connection. The multimeter needle should not move at all. If all the readings are HIGH, progress to the Z80, and then each of the other chips. If the pointer deflects at any stage, trace through the wiring to see if a resistor or push button is in the path. You will also get some low readings when testing near the display. So don't treat these as faults.

USE THE LOGIC PROBE AS DESCRIBED IN THE FIRST PROJECT, TO TEST THE TEC-1.

Another very important check you can make is the continuity of all the printed wiring on the underside of the board. Sometimes one of these tracks can become eaten away in the etching process, resulting in a break.

Place one of the probes on one end of a conductor and visually trace it through to the end. Place the other probe at this point and prove that it is conducting. You can also make sure the jumpers are connecting by checking the ends of each run.

If all these checks fail to locate any problem, you will have to carry out tests with the computer operating. This will mean replacing the chips and connecting the power.

Start by placing the probe on pin 6 of the Z80. This is the clock input pin and without any signal here, the whole computer will not operate. The three LEDs on our LOGIC PROBE will illuminate and you will hear a frequency from the mini speaker in the probe. As you adjust the speed control, the sound will change pitch. If the 3 LEDs don't flash, change the 4049. Some chips are very critical in this circuit and others don't work at all. If a chip fails to oscillate, you can reduce the 10k to 2k2 and this will give you a broader range. Some chips may tend to drop out at the low end. You should buy a CD 4049 as soon as possible.

Once you have a clock pulse entering the Z80, you can check some of the other sections of the computer.

The computer can be placed in a WAIT situation by tying pin 24 to earth. This pin is connected to the 10k which is the closest to the reset switch. It has an empty hole to which you can solder a test wire and use a jumper lead to create the wait situation. Press RESET and probe pin 15. If it is LOW, everything is OK. If it is not LOW, you may have a dry joint or short-circuit on pins 1 - 6 of the 74LS138. While the computer is in the WAIT condition, test the operation of the keyboard by probing pin 15 of the 4049. This is the output of the 74C923 key-board encoder, after it has passed through an inverter to the non-maskable interrupt of the Z80.

This output line is normally HIGH and goes LOW when a key is pressed. The only other pin of the 74C923 which can be checked in a simple manner is pin 7. It is normally LOW and goes HIGH for the duration a key is pressed.

The 74C138 select chip below the expansion port selects between the keyboard and the two 8212 driver chips. When in the wait mode, pins 13, 14 and 15 are HIGH. Under running conditions, pin 15 pulses LOW when a key is pressed.

The two display driver chips, (8212) are difficult to test under static conditions as the display is multiplexed.

When in the wait mode, one of the displays may illuminate and you will be able to detect the HIGH's entering the 8212's.

The advantage of the sockets becomes apparent when you have to remove or change any of the chips.

If the computer still fails to operate correctly, try replacing the set of chips. This will only be feasible if you know someone with a TEC-1. Within your own board you can exchange the two 8212's and 74LS138's. Don't forget, the 2716 must be programmed. A blank one will not get the computer started.

Make sure no pins are bent under the sockets or broken off at the chip. Be sure the chips are around the correct way and most of all, make sure the chips are in the correct positions.

If all this fails, write to us. We have a repair service available. If sending the project by post, pack the board between two thick pieces of foam or stiff cardboard. Use a large jiffy bag and mark it fragile. Certify the parcel in case anything is damaged. This way it will get to us in one piece. We will have a look at your project and let you know what it will cost to fix. Usually it doesn't cost much and this will take a load of your mind.

I hope it never gets to this stage, but at least you know the service is available.

EXPERIMENTS FOR THE TEC-1

The computer should now be fully assembled and ready to go. All you have to do is learn how to operate it.

The following set of experiments will give you the experience necessary to recall some of its routines and produce a simple sequence of your own.

To introduce you to the TEC-1 we have programmed a welcome message into the EPROM. This can be located at 02D1.

To call up this program, press the following sequence of keys:

RESET, D, 1, + 0, 2, ADdress, 0, 2, 7, 0, GO, GO.

Adjust the speed control and see what John has written. If that's not a clever way of personalizing a piece of equipment!!

If you don't know what to do, don't worry. Follow through these experiments and come back to the WELCOME mesage later.

Now to the learning section:

Experiment 1.

AIM: To examine the increment of the address.

Apparatus:TEC-1.

Procedure: Turn the TEC-1 on and look at the address display. The address is the first four digits. When the TEC-1 is turned on, the first available address is 0800. This can be incremented by pressing the '+' key and the address will increase to the next available location. Carry out this procedure by pressing the '+' key and watch the display: 01, 02, 03, 04, 05, 06, 07, 08, 09. The next location is 0A and this is where the computer departs from the reading you would expect. The TEC-1 is programmed in Hexadecimal in which 4 binary lines are grouped together to form a hex number. In this way we can write binary numbers from 0000 to 1111 and this means we can go higher than 9 as nine is only 1001 in binary. The next hex number is A, then B, C, D, E, and finally F. The following table shows the binary equivalent for 0 - F.

Decimal:	Hex:	Binary:
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

A comparison between decimal numbers (based on the power 10) hexadecimal numbers (based on the power 16) and binary numbers (based on the power 2). The data for TEC-1 is entered in hex on the key-board. Examples of hex are: 3F, 4C, 5B, FE, C4, DD,
The max hex for 2 digits is FF and this corresponds to 255.

Press the + and watch the display increment,
Press the − key and watch the display decrement.

Experiment 2
STUDYING HEX

Aim: To study hex notation and count in Hex.

Equipment: TEC-1.

Theory: Each byte of data for a program must be given an address. The computer automatically advances one address location on pressing the + key. However we must be able to read and write hex values to be able to prepare a program.

Procedure: Study the Hex notation in expt 1. and answer the following set of problems:
Use the TEC-1 to verify your answers.
Problem 1: A program starts at 0800. What are the next 21 addresses?

0801 ★ ★ 0804 ★ ★ ★
0808 ★ 080A ★ ★ ★ ★
★ ★ 0811 ★ ★ ★ 0815.
To verify your answer, press RESET then keep pressing + + + + + etc. Don't worry about the values in the data displays.

Problem 2: A program starts at 0A00. Complete the following set of addresses:
0A00 ★ ★ ★ ★ ★ 0A06
★ ★ ★ ★ 0A0B ★ ★ ★
★ ★ ★ 0A12.
To locate address 0A00: Press RESET, press AD (the dots will appear on the address displays indicating the address can be changed).

Press 0, A, 0, 0. Press +. This becomes the first address location. Press + + + + + etc to increment the display.

Problem 3: A program of 50 addresses finishes at 091E. What are the previous 35 addresses?
091E ★ 091C ★ ★ ★ ★
★ ★ etc to 08Fb.

Problem 4: (a) Add 4 address locations to 0209.
(b) Add 8 address locations to 1FFF.
(c) Add 4 address locations to 0BFD.
(d) Decrement the address 7 locations from 0800.

Work out all the above answers on paper before checking with the TEC-1

ANS: 4(a) 020D, (b) 2007 (c) 0C01 (d) 07F9

Experiment 3:

CREATING A BEEP

AIM: To create a tone or beep on the TEC-1.

Theory: The 2716 has been pre-programmed with a loop to give a pulse to the speaker. Depending on the speed of the system, the tone of the pulse will be varied.

Procedure: We can address the beginning of this routine by pressing the following keys:
RESET, ADdress, 0, 1, 8, E, GO, GO.

You will hear two beeps, and by turning the speed control down, they will become separated. The first beep is the one you have programmed. In the next experiment you will will change the frequency of the beep.

Notes:
When the TEC-1 is reset, the decimal points appear in the DATA readouts. This indicates the data can be changed by pressing the keys 0 - F.

By pressing the ADdress key, the dots will appear in the ADDRESS readouts. This can now be changes by pressing the keys 0 - F.

Experiment 4:

Creating a Tone or Note.

AIM: To create a tone.

Theory: It is possible to produce a tone from the speaker which is the result of a routine in the EPROM.

Procedure: To create a single note or tone, press the following:

RESET, 2 + 8 + **1** *ADdress, 1, B, 0, GO, GO.*

Only the first note to be heard in the speaker is the product of our programming. The other beep or beeps come from other routines.

The number **1** in the routine above selects the particular note. It can have one of 24 different values and thus we can create different effects as shown in a later experiment.

AIM: To create a single note with a period of silence.

The instruction for silence is 00.

Run the following program:

2+8+0,3+00+00+00+00+0,3 + 00 + 00 ADdress 1, B, 0, GO, GO.

You will notice that pressing 00 is the same as pressing 0. The DATA entry is self-adjusting. Thus 03 is the same as 3 on the data display.
Turn the TEC-1 off between experiments so that the 6116 RAM has its contents destroyed. Otherwise some of the previous programs will come through the speaker.

To create a scale:

Program this sequence:

2+8+1+2+3+4+5+6+7+8 +9+A+B+C+D+E+F+10+ 11 + 12 + 13 + 14 + 15 + 16 + 17 + 18 + 0 + 0 + 0 + 0 Address 1, B, 0, GO, GO.

The speaker will produce the scale, then silence.
To hear the sequence again: Press RESET, ADdress, 1, B, 0, GO, GO.

AIM: To produce a repeat function.

The note, notes or sequence can be repeated by adding the instruction 1E to the end of the list.

Try this routine:

2+8+1+2+3+4+5+6+5+4 +3+2+1+1,E Address 1, B, 0, Go, Go.

To produce a repeat function with a pause or silence, try this routine:

2+8+1+2+3+4+5+0+0+0 +0+5+4+3+2+1+1,E Address 1, B, 0, GO, GO.

Experiment 5:

To Create A Tune.

By using the note table on P.73, any tune or melody can be produced. Try this sequence, then write your own tune.
2+8+0A+08+06+08+0A+0F +0A+0D+0F+06+06+0A+0D +06+0D+0A+0D+12+16+14 + 12+0f+11+12+0F+0D+0D +0D+0A+12+0F+0D+0A+08 +06+08+0A+06+06+00+1,E, Address 1, B, 0, GO, GO.

Question: How do you recall this sequence?

SOUNDS AND TUNES

The TEC-1 has a number of musical routines programmed into the 2716 EPROM.

These are accessible via the keyboard. The first of these is an Irish Jig. This is called by the sequence:

RESET, E, F, GO,

With all tunes the pitch of the notes is dependent upon the speed of the computer. This is determined by the speed control.

You can experiment with adjusting the 20k cermet for each of these tunes, to get different effects.

In order to access some of the other tunes, you will need to follow this key sequence:

RESET 3 0 + 5 address 1 B 0 GO, GO.

Experiment 6:
Creating a Running Letter

AIM: To produce a running A.

Procedure: Press the following sequence of keys:

RESET, 2 + 8 + 1 + 0 + 0 + 0 + 1,E Address 2, 7, 0, GO, GO.

The letter A is being shifted one place to the left by the routine at 0270.

The letter can be made to travel the full length of the display by adding further zeros to the program.

Press this sequence of keys:

RESET 2 + 8 + 1 + 0 + 0 + 0 + 0 + 0 + 0 + 1,E address 2, 7, 0, GO,GO.

To produce a running sentence:

Press this sequence of keys:

RESET 2 + 8 + 07 + 0E + 0E + 04 + 0 + 9 + 04 + 05 + 01 + 1A + 0 + 0 + 0 + 0 + 0 + 1,E address 2, 7, 0, GO, GO.

Experiment 7:
Combining Words With A Tune

AIM: To combine a tune and running words in one sequence.

Procedure: The programme we will be writing in this experiment consists of a set of instructions and included in this are two CALL statements (call 1B0 and call 270) followed by a PITCH table and a LETTER table.

The result of your programming will be a short tune followed by three letters running across the screen and this will be repeated.

We will firstly describe the program for experiment 7 in **WORDS.** Refer to the program below to see what we are talking about.

The program starts at 0800 and in the first two bytes (800 will accept a byte of data and 801 will accept a byte of data) we will store the address of the pitch table (which starts at 0900) . Later the computer will store the letter table and this will be repeated over and over again as the program contains a jump or repeat instruction.

First of all we will discuss each instruction at each address so that you will be able to understand the program you will be keying into the TEC-1.

At address 802 the instruction 3E tells the accumulator to load the immediate byte, which is 00.
This takes up address 802 and 803.

At address 804 the instruction 32 tells the computer to load the contents of the accumulator into the address given by the following two bytes. The lowest byte is always presented first, then the highest-order byte. Thus 00 is loaded first then 08.

The next available address is 807.

The instruction 3E tells the accumulator to load with the immediate byte which is 09.

At address 809 the instruction 32 tells the computer to load the contents of the accumulator into the address given by the following two bytes.

Address 80C. This is a CALL instruction which calls the routine located at 1B0. This is the address of the music programme.

At 80F the instruction is to load the accumulator with the contents of the immediate byte which contains 0A. This 0A is the most significant byte of the address for the letter table. As 800 already contains the byte 00 (the least significant byte of the address for the letter table) we do not have to load it again.

At address 811 the instruction is to load the address 801 with the contents of the accumulator (800 is already loaded correctly).

At address 814, the instruction is to call the letter printing routine located at 270. This routine contains an instruction to look at location 800 and 801 and see where the look-up table is located. In our case it is at 0A00.

The final address 817 is an instruction to JUMP to address 802. This is used as a repeat function.

THE PROGRAM:

```
800   Push + + to get 802.
802   LD  A,00          3E  00
804   LD  (800), A      32  00  08
807   LD  A, 09         3e  09
809   LD  (801), A      32  01  08
80C   CALL  1B0         CD  B0  01
80F   LD  A, 0A         3E  0A
811   LD  (801), A      32  01  08
814   CALL  270         CD  70  02
817   JP  802           C3  02  08
```

The first column is the address of the memory location in RAM.

The centre column is the assembly language in mnemonics.

The third column is the Machine Code listing.

PITCH TABLE:

0900:

```
01
00
01
00
02
03
04
05
04
05
1F - means to return to
        line LD A, 0A.
```

LETTER TABLE:

0A00:

```
01
02
03
00
00
00
00
00
00
1F - means to return to
        line JP 802.
```

To run the program: Press: Reset, +, + ,GO, GO.

This is how the sequence should be keyed:

Press RESET to get the first location. Press: *00 + 09 + 3E + 00 + 32 + 00 + 08 + 3E + 09 + 32 + 01 + 08 + CD + B0 + 01 + 3E + 0A + 32 + 01 + 08 + CD + 70 + 02 + C3 + 02 + 08 ADdress 0900 + 01 + 00 + 01 + 00 + 02 + 03 + 04 + 05 + 04 + 05 + 1F ADdress 0A00 + 01 + 02 + 03 + 00 + 00 + 00 + 00 + 00 + 00 + 1F RESET + + GO,*

You now have enough information to be able to produce your own sequence. Try a longer note sequence and a longer sentence. You have the availability of including 255 in each table.

The next issue of TE will show how to write a program to display ONE segment of any particular digit, then two segments, so that you can create your own characters.

This is the beginning to writing programmes for video games and you will be shown how to prepare the internal structure of a simple moving target game.

We will also introduce some expansion and interface projects. So, be prepared.

Use the following table to create your own tunes.

NOTE TABLE

G	01
G#	02
A	03
A#	04
B	05
C	06
C#	07
D	08
D#	09
E	0A
F	0B
F#	0C
G	0D
G#	0E
A	0F
A#	10
B	11
C	12
C#	13
D	14
D#	15
E	16
F	17
F#	18
Repeat	1E
Return	1F

Use this table to create your own words:

LETTER TABLE

	00
A	01
B	02
C	03
D	04
E	05
F	06
G	07
H	08
I	09
J	0A
K	0B
L	0C
M	
N	0D
O	0E
P	0F
Q	10
R	11
S	12
T	13
U	14
V	15
W	
X	
Y	16
Z	17
–	18
.	19
!	1A
Repeat	1E
Return	1F

NIM

Key sequence: AD, 3, E, 0, GO,GO.

When the game ends, press any key to restart.

You are playing against the computer in a battle of wits. The computer has an obvious advantage.

There are 23 matches and you take turns in removing 1, 2, or 3 matches. The object of the game is to make the computer take the last match.

At each turn you can only take 1, 2, or 3. The computer lets you go first. The number of matches is displayed on the last two digits of the display. When you press a button, say 3, this will be displayed as **Y 3.** This indicates you took 3 matches. Then it will display **I 3.** This will mean the computer took 3 matches.

It is now waiting for your next move. Be careful, the computer is smart. It is waiting for you to make your first mistake. It will then take immediate advantage of it.

If you are playing a purely random game, you will find yourself holding the last match every time. The computer will let you know too! Read the message it displays!

The computer is a pretty bad loser but be thankful it doesn't self-destruct in disgust!

If you are playing a careful **well-calculated** game, you can **WIN.** So, try your skill and see the winning message.

LUNA LANDER

Key sequence: AD, 4, 9, 0, GO, GO.

Set speed control to your level of skill (strength of gravity). When the game ends, press any key to restart.

You are in a luna module, orbiting some 50 kilometres above the luna surface. You have 20 litres of astro fuel left and you have to land your spacecraft without denting either the moon or the craft.

Gravity is constantly pulling you down and you can only slow your descent by blasting with your retro rockets.

Your height is indicated by the first two digits and this starts at 50. Watch yourself descend without blasting your retros and as you fall, you will descend faster and faster - until you HIT!

Press any key to restart (except reset). To blast for a short time, press: +. This may slow you a bit and to slow yourself down more, press + several times. If you over-do this command, you will slow down to zero velocity and even start going UP! Never move upwards as this is a waste of fuel.

Every time you blast, your fuel goes down by ONE LITRE. Once your fuel runs out, you can't fire any more and you start falling towards the luna surface.

So, use your fuel wisely to survive!

INVADERS

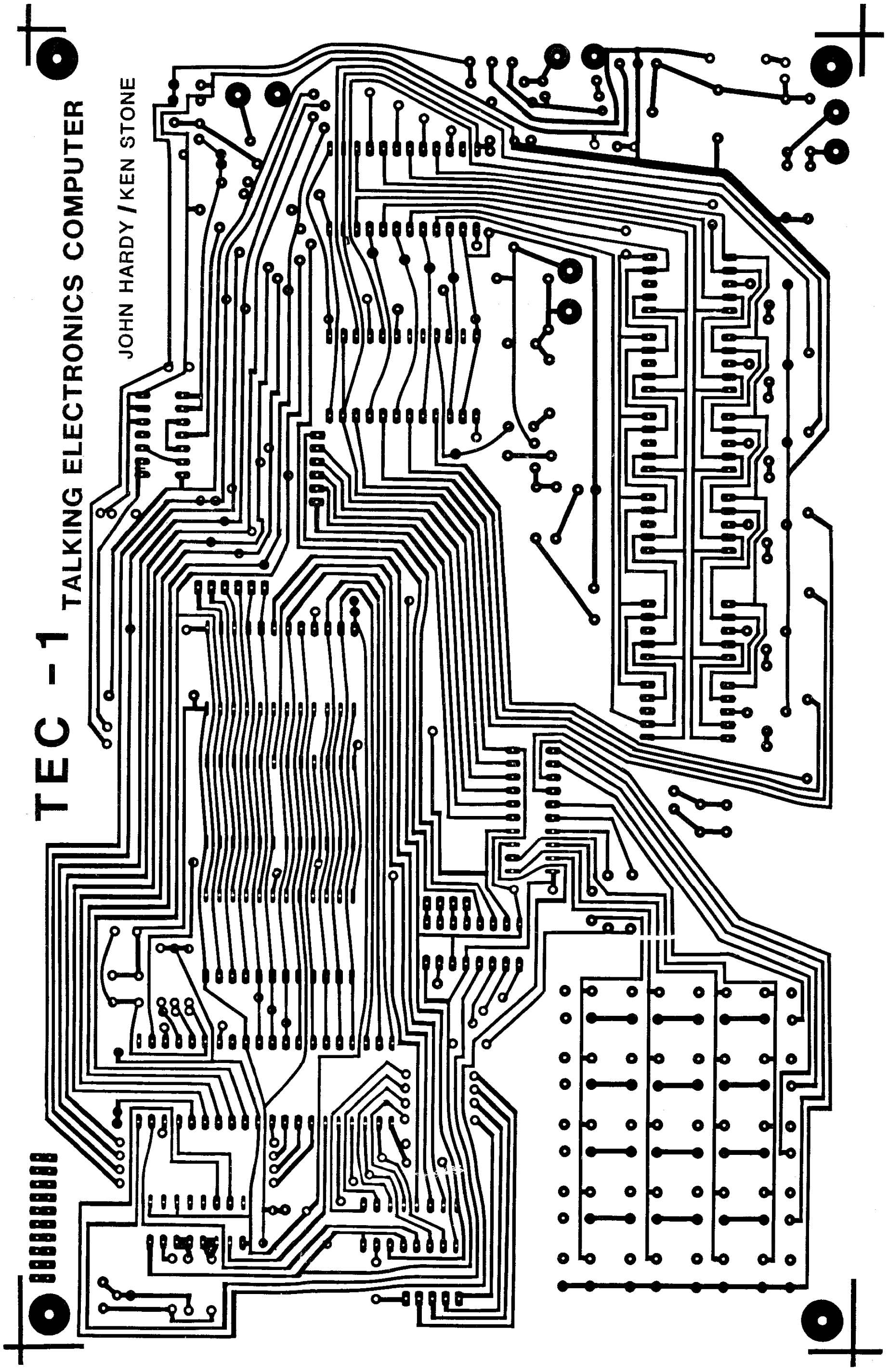
Key sequence: AD, 3, 2, 0, GO, GO.

Set speed control to your level of skill. When the game ends, press any key to restart.

The object of **INVADERS** is very simple. You shoot anything that moves! Your position is represented by the number on the left. The invaders appear from the right. They shift across the display and if they touch you - "POW". The game ends and the score is shown on the screen.

You can't stop the invaders advancing but you can defend yourself by blowing them up. The fire button is button 0 but you can only destroy those invaders which have the same number as your space-gun.

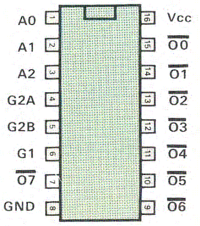
To change your number to match the first invader, press the + button. You can only increase your number and not reduce it. By using the + and fire keys you will be able to keep the invaders at bay. To improve your skill, advance the speed control. You can also destroy those behind the front invader by matching the numbers.
Try your fire power, you'll find it most absorbing.



TEC -1
TALKING ELECTRONICS COMPUTER
JOHN HARDY / KEN STONE

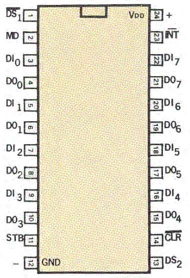
DATA SHEET 10

74LS138



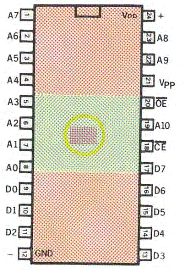
1 - 8 DECODER

8212



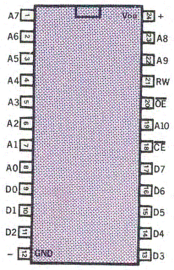
LATCH

2716



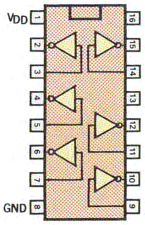
2k x 8 BIT EPROM

6116



2k x 8 BIT RAM

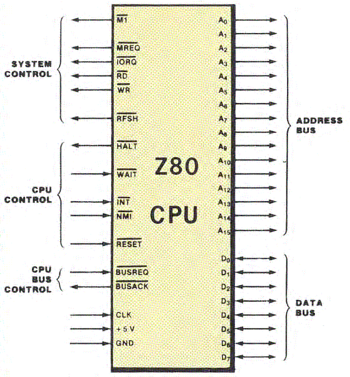
4049



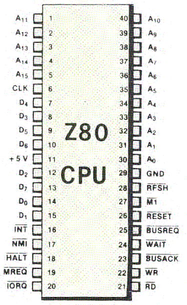
HEX INVERTER

Z80 CPU

SOURCE: SGS DATA BOOKS.

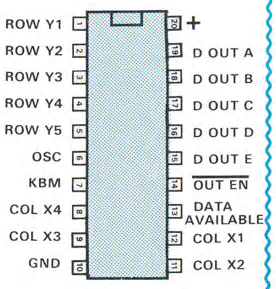


Z80 LOGIC FUNCTIONS

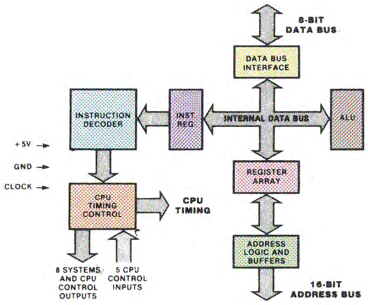


Z80 PIN OUTS

74C923



20-KEY ENCODER



Z80CPU BLOCK DIAGRAM

DESIGNED and PUBLISHED by COLIN MITCHELL at 35 Rosewarne Ave., CHELTENHAM, VICTORIA, 3192 (03) 584 2386

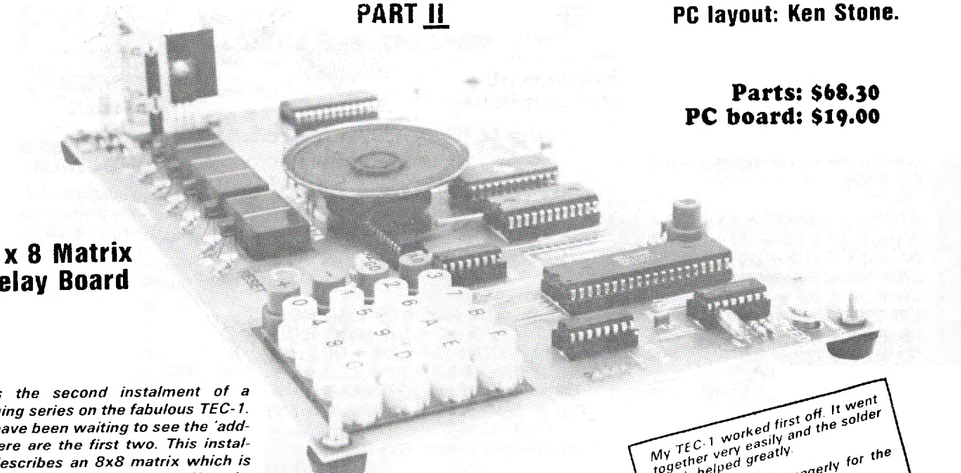
TEC-1

TALKING ELECTRONICS COMPUTER

PART II

–by John Hardy
PC layout: Ken Stone.

Parts: $68.30
PC board: $19.00



★ **8 x 8 Matrix**
★ **Relay Board**

This is the second instalment of a continuing series on the fabulous TEC-1. If you have been waiting to see the 'add-ons', here are the first two. This instalmant describes an 8x8 matrix which is effectively a WINDOW ON A VDU, and a RELAY BOARD which contains a set of 8 relays so that the TEC-1 will access the outside world.

You can operate globes or motors via the relays or drive them directly via the set of transistors included on the board.

The 8x8 matrix is multiplexed and driven by its own set of latches. In the ultimate you will be able to get incredible movement, but in the elementary stage its simple illumination and shift patterns.

Now that you have got this far, read on.. .. .. ..

The introduction of the TEC-1 in the previous issue caused quite a lot of interest from a new group of hobbyists. Was this due to the colour cover or the presentation of a cheap computer? Who knows?

In any case, we are pleased it took their attention. Everyone will benefit with the increased sales it produced.

We noted the number of orders increased dramatically with many coming from names and places not on our mailing list or files.

The requests for TEC-1 outstripped the availability of kits and we soon realized the small markets in Australia had to be by-passed in preference to direct importing.

Sales are still peaking but I think many readers are still waiting for the full range of "add-ons" before launching into purchasing a kit. Let's hope some of your answers will be answered when you see the extent of the projects in this issue. And this is only just the beginning.

We have already designed more than 9 different expansions for the TEC-1 and this will take it into the field of a fully-fledged demonstrator.

Within the first week we received 5 letters from constructors who had the TEC-1 operating from the instant of switch-on.

Although extremely simple, the TEC-1 works very well. Some of its features are novel while others are a little outdated. The speed control is a novelty while the 8212's have been around for years and are now getting towards the end of production. We found this out as they are now quite

. . .cont. P. 14.

My TEC-1 worked first off. It went together very easily and the solder mask helped greatly.

I am now waiting eagerly for the next installment.

Steven Truscott,
2287

When the TEC-1 was introduced in issue 10, my son and I agreed it should be a good place to learn about computers. We built the kit and it worked straight away. We were quite impressed by the quality of the PC board and the technical details in issue 10.

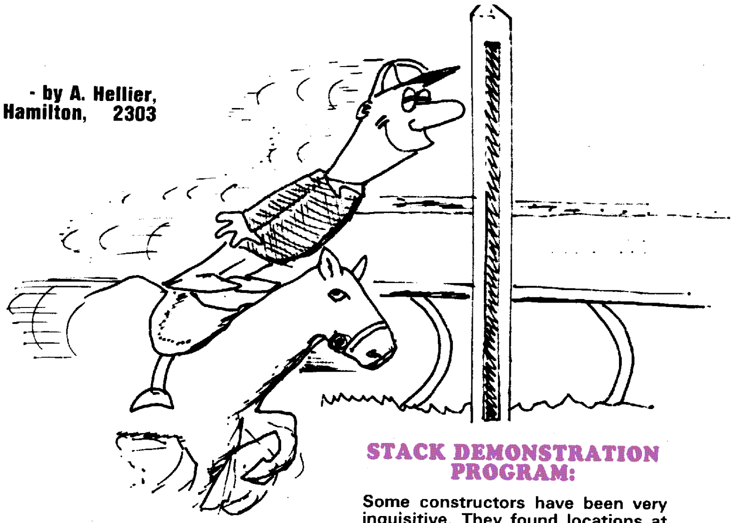
We are now in the process of making a case for the TEC-1 and are in complete agreement that the computer should be exposed so that we keep in touch with the "operations". We will be fitting a hinged perspex cover to keep out dust etc. The only problem is the heatsink on the 7805. We have decided to mount the regulator under the board, near one corner and run three lines to the appropriate lands. Everything will then be neat, firm and tamper-proof.

Martin Hulsman,
7310.

. . . . .In the expansion port on the TEC-1 I would mount one of those IC sockets with a little lever on it. They are expensive but make it easier to remove the expansion plugs.

Raymond Green.

- by A. Hellier,
Hamilton, 2303



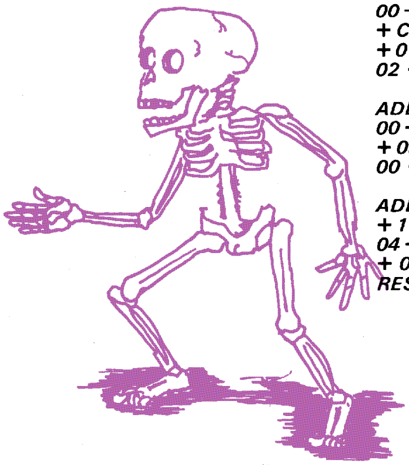
BIG BEN CHIMES

```
RESET 00 + 09 + 3E + 00 + 32 +
00 + 08 + 3E + 09 + 32 + 01 + 08
+ CD + BO + 01 + 3E + 0A + 32 +
01 + 08 + CD + 70 + 02 + C3 +
02 + 08.

ADDRESS: 0900 + 11 + OD + OF
+ 08 + 00 + 00 + 00 + 00 + 08 +
OF + 11 + OD + 00 + 00 + 00 +
00 + 11 + OF + OD + 08 + 00 +
00 + 00 + 00 + 08 + OF + 11 +
OD + 00 + 00 + 00 + 00 + OD +
00 + 00 + 00 + 00 + OD + 00 +
00 + 00 + 00 + OD + 00 + 00 +
00 + 00 OD + 1F.

ADDRESS OAOO + 02 + 09 + 07
+ 00 + 02 + 05 + OD + 00 + 00 +
00 + 00 + 00 + 00 + 00 + 1F.

RESET + + GO GO.
```

WINNERS CALL

```
RESET + 02 + 08 + 06 + 0A + OD
+ 12 + 12 + 12 + OD + OD + OD
+ OA + OD + OA + 06 + 00 + 06
+ OA + OD + 12 + 12 + 12 + 06 +
06 + 06 + OD + OD + OD + 00 +
00 + 1E.

ADDRESS 01B0 GO GO.
```

"YOUR'RE DEAD" FUNERAL DIRGE



```
RESET 00 + 09 + 3E + 00 + 32 +
00 + 08 + 3E + 09 + 32 + 01 + 08
+ CD + BO + 01 + 3E + OA + 32
+ 01 + 08 + CD + 70 + 02 + C3 +
02 + 08.

ADDRESS 0900 + 03 + 00 + 03 +
00 + 03 + 03 + 00 + 06 + 00 + 05
+ 05 + 00 + 03 + 00 + 02 + 03 +
00 + 1F.

ADDRESS OAOO + 16 + OE + 14
+ 11 + 05 + 00 + 04 + 05 + 01 +
04 + 1A + 1A + 00 + 00 + 00 + 00
+ 00 + 1F.
RESET + + GO GO.
```

STACK DEMONSTRATION PROGRAM:

Some constructors have been very inquisitive. They found locations at the high end of RAM which they could not remove! (This is because the TEC-1 was replacing them again).This was quite puzzling as we know anything in RAM can be removed and written over.

But this area is special and is called the STACK area.

When a PUSH instruction is executed by the Z80, the contents of the location are loaded into this area. The stack starts at OFFO for MON-1A EPROMS and OFDO for MON-1B EPROMS. It advances downwards, towards the LOW addresses in the 6116.

Each time a POP (or PULL) instruction is executed, the last item to put onto the stack is removed and the stack gets smaller. Otherwise it gets bigger and bigger.

If a program contains too many PUSH instructions, the stack will grow and eventually hit the program. This will make the computer CRASH!

The simple program below pushes register pair (they must be in pairs) **AF** into the stack and completely fills the RAM.

Try the program and watch the computer CRASH.

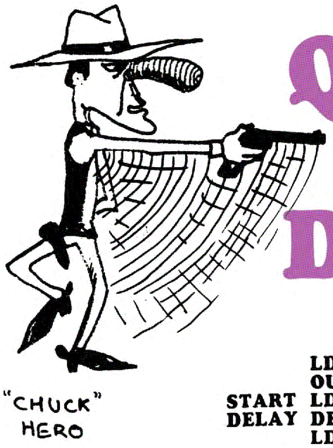
```
11 AA BB
D5
C3 03 08
```

RESET, GO.

Increment the address and prove the 6116 is completely filled with AA, BB, AA, BB etc.

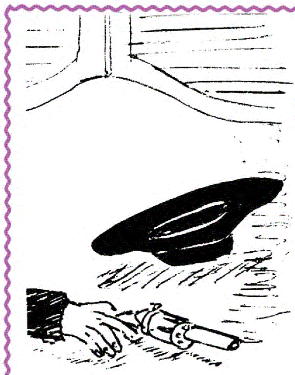
Change AA, BB to CC, DD and repeat. Check the RAM and read its contents.

QUICK DRAW



THE PROGRAM

```
        LD A,00         800   3E 00
        OUT(1),A        802   D3 01
START   LD DE,00        804   11 00 00
DELAY   DEC DE          807   1B
        LD A,D          808   7A
        OR E            809   B3
        JP NZ Delay     80A   C2 07 08
        LD A,E3         80D   3E E3
        OUT (2),A       80F   D3 02
        LD A,08         811   3E 08
        OUT (1),A       813   D3 01
LOOP 1  HALT            815   76
        AND OF          816   E6 0F
        CP OC           818   FE 0C
        JP Z,Right      81A   CA 24 08
        OR A            81D   B7
        JP Z,Left       81E   CA 29 08
        JP Loop 1       821   C3 15 08
RIGHT   LD A,01         824   3E 01
        JP Finish       826   C3 2B 08
        LD A,20         829   3E 20
        OUT (1),A       82B   D3 01
LEFT    LD A,20         82D   3E 28
FINISH  OUT (2),A       82F   D3 02
        HALT            831   76
        JP Start        832   C3 00 08
```



QUICK DRAW is a reaction game for two players.

To start the game, press RESET, GO.

After a DELAY, as determined by the delay routine at 804, the letter G will appear on the screen. The first player to press his button will be detected by the computer and result in the figure 1 appearing on the appropriate end of the display.

Player 1 uses the + button and player 2 uses the 'C' button.

Any button can be pressed to reset the game.

The first instruction is to load the accumulator with zero and output this to port 1 to prevent odd segments lighting up when the game is reset.

At address 804, the register pair DE is loaded with the value 00, 00. Surprisingly, this creates the longest delay as the first operation in the delay routine is to decrease the lower byte (register E) by one. This immediately removes the value of zero from the pair and when D is loaded into the accumulator, and the logical OR operation performed, the answer will only be zero when both the accumulator and register E are completely zero.

If one or both are not zero, the program will jump to instruction 807 whereupon register pair DE will be decremented by ONE. This loop will be cycled 256 x 256 times and each time will occupy quite a number of machine cycles.

This results in the letter G (for GO) taking a few seconds to appear on the screen. This creates the same effect as the delay circuit in the Quick Draw project described in issue 5.

When the register pair becomes zero, the program is advanced one address and the accumulator is loaded with the value E3. The value E3 will produce the letter G on the screen and the location of this letter is determined by loading the accumulator with 8 and outputting it to port 1.

The computer is now HALTED and waits for an input instruction. If any of the keys are pressed, the 74c923 will activate the NON-MARSKABLE INTERRUPT line and present data to the Z80 according to the value of the key.

If key C is pressed, the value 1010 is placed in the accumulator. This is then logically ANDed with the value F (1111) and the result appears in the accumulator.

When a number O-F is ANDed with 1111, the answer will be exactly the same as the number itself. In actual fact, this AND 0F operation is not required for the jump right command and you can ignore it.

After the AND OF operation, the number we are looking for is 1010 (for a jump RIGHT). The answer is compared with OC and the Z80 does this by effectively performing a subtraction operation in which the value C is subtracted from the contents of the accumulator.

If the answer is zero, the computer is instructed to jump to address 824. If the answer is not zero, a logical OR operation is performed with the accumulator as the operator and also the operand.

Since the accumuator is zero, the answer will be zero. Thus the program will advance via a jump instruction, to 829. If neither of the conditions are met, the CPU will jump to 815 and wait for a key to be pressed.

If the processor advances to location 824, the accumulator will be loaded with the value 1 and told to jump to address 82B. This value 1 is outputed to Port 1 and sets one of the latches ready to display the far right-hand digit.

At address 82D, the accumulator is loaded with 28 so that the segments 'a' and 'b' will illuminate when the value 28 (not twenty eight but two, eight) is outputed to Port 2.

Finally the processor is told to HALT at address 831.

On the other hand, if the program advances to 81E, the processor is told to jump to address 829 where the accumulator is loaded with 20 so that the far left-hand display will be activated when port 2 is given the value 20.

The program can be re-started by pressing ANY key, and the accumulator is loaded with zero at 800 so that odd characters do not appear on the screen.

Here are 6 simple experiments which can be performed on this program to better understand how it operates:

1. Load address 801, 2, 3 and 4 with 00 and play the game a few times. Notice how odd figures appear on the screen. Replace the correct program values and continue:
2. At address 812, load the value 10, or 04 or OF and note the different effects.
3. At address 816 and 817, insert 00 00. What effect does this have?
4. At address 819, insert OD or OE or OF. What is the result?
5. At address 829, load the value 10 and see the result.
6. Finally, insert the value 01 at location 806. Try the value 06, 0A BB or FF. What effect do they have?

difficult to obtain. The other chip on short-supply is the 74c923 as it is only made by one manufacturer.

The FND 500 or FND 560 displays are no longer in production by Fairchild (as they have ceased to produce OPTO devices) however other suppliers have produced identical replacements.

Apart from this, the TEC-1 is straightforward.

Out of the first 300 kits we had reports from only 6 readers who had trouble in getting the computer to work.

Paul had incorrectly placed the "six" button in the keyboard so that the flat was 90° out of position. The TEC-1 came on at address 0800 but the keyboard did not operate.

The 74c923 detected key 6 as being pressed due to the wiring of the contacts and the NMI being activated.

We traced the fault by removing the 74c923 and checking pins 8, 9, 11, 12 and 1, 2, 3, 4, 5 with an ohm-meter. The short between pins 2 and 9 showed key 6 to be at fault and that was when we noticed it!

Another TEC-1 came in with a very faulty + button and a broken PC line under one of the 7-segment displays.

Two computers had shorts between tracks around the memory section where the tracks are very close to one another.

But possibly the worst effort came in the post last week. The 74c923 socket had been made up with a 14 pin and an 8 pin so that 2 pins projected too far. The chip has been inserted so that pin 2 connected with pin 1 on the circuit. The other fault was a jumper missing near the first 8212. Both of these faults showed lack of inspection. When they were repaired, the computer worked perfectly.

Finally a constructor arrived with a TEC-1 under his arm. It gave an occasional display of odd segments and a beep from the speaker when switched on. The trouble was traced to a faulty Z80!

Apart from the above cases, the TEC-1 seems to offer a high degree of success.

If you have any problems with your unit, let us know. We want to present them in the next issue, under FAULT FINDING.

The introduction of the TEC-1 is primarily intended to unlock three areas of microprocessor design. These are:

1. To teach Machine Code programming.
2. To teach the art of creating Video Games and,

3. To access the real world.

MACHINE CODE programming is the skill of telling a microprocessor what to do by writing directly into a memory bank. The memory can either be a tempory storage (RAM) or permanent storage ROM. The main difference between these two is ROM will retain its memory when the power is turned off whereas RAM will lose its contents.

We can use this feature to write into RAM and then erase the program by turning off the power. Alternatively we can simply write over the old program. Both ROM and RAM are used in the TEC-1.

The Z80 has the capability of accepting over 700 instructions, some of which are 2-byte. These take the form of a two digit number which is written in hexadecimal form. The first large table in this installment shows how to write any of these numbers and explains how we arrive at C2 or E5 or D7 as a value in this instruction set.

A typical instruction is 3E. This means "load the accumulator (register A) with the following value . . ." Once you remember some of these instructions you will see why we have concentrated on Machine Code Programming.

Instructions such as 76 for HALT, C9 for RETURN and C2 for JUMP NOT ZERO are quite easy to remember. The meaning of JUMP NOT ZERO needs a little explanation. After C2 you must insert 2 bytes because the computer will interpret whatever is placed as the next two bytes as an address location. For example, 20 08 will tell the processor to jump to 0820. The instruction C2 also infers that if the program IS ZERO, the processor will proceed to the next instruction.

Machine Code is the only approach for video game development. It produces the fast-moving games as seen in the latest coin-in-the-slot machines. Any of the games on cassette are theoretically possible with the TEC-1. Mind you, we will not be advancing to the complexity of colour or the swirling action of VENUS, but in a developmental way you will have the opportunity to program sections of a video game and watch the result.

PROGRAMMING STARTS HERE:

By now you will have completed the first 7 experiments and possibly tried some additional programming of your own.

But just in case you did not absorb all the facts, we will go through some of the experiments again to make sure everything is understood.

Just before we start, key the following:
Reset, two, +, eight, +, two, +, five, +, fourteen, + four, +, sixteen, +, 1A, +, zero, +, zero, +, zero, + zero, +, zero, +, 1E, ADdress 2, 7, 0, GO, GO.

DO YOU AGREE?

You should remember a few simple programs like this, to impress your friends.

The Z80 instruction-set has 8 special restart instructions which are single-byte instructions. By keying in one of the following: C7, CF, D7, DF, E7, EF, F7, or FF the computer will go to address location 00 or 08, 10, 18, 20, 28, 30, 38 and will receive information to go to the beginning of the 2 tunes and 3 games.

The 2 tunes are accessed by EF and F7. Push Reset, EF, GO. EF tells the Z80 to go to location 28 in the EPROM. At this address is an instruction 21 which tells the Z80 to load register pair HL with the contents of another location which is the beginning of the song table.

F7 directs the Z80 to go to location 30 and this address directs the Z80 to another song table.

We do not use C7, E7 or FF but the other 3: CF, D7 and DF select the commencement of the three games.

Try them.

When the computer is turned on, or the reset button pressed, the first available address is 0800.

This means the address locations from zero to 0800 have been allocated to the 2716 EPROM and most of this has been filled with start-up instructions and games. This includes a music table and letter table which are user accessible and a number table which is only computer accessible.

Actually the MON-1 EPROM has been filled from zero to 05A7, and we will now look at how many locations this represents.

LEARNING HEX

We know the TEC-1 is programmed in hexadecimal. This means the locations commence at 0000 and increase: 0001, 0002, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 2A, 2B, 2C, 2D, 2E, 2F, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 3A, 3B, 3C . . . . . .9D, 9E, 9F, A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, AA, AB, AC, AD, AE, AF, B0, B1, B2, B3, etc up to 00F2, 00F3,00F4,00F5,00F6, 00F7, 00F8, F9, FA, FB, FC, FD, 00FE, 00FF.

If you fill in all the blanks between 0000 and 00FF, you will find there are 16 lots of 16 addresses. This is equal to 256 locations.

So, between 0000 and 00FF there are 256 address locations.

The next location 0100 (it is best to say oh, one oh, oh as the location is not really one hundred). Between 0100 and 01FF there are another 256 locations.
Between 0200 and 02FF there are another 256 locations.
0300 - 03FF = 256 locations
0400 - 04FF = 256 locations
0500 - 05A7 there are ??? locations.

Let's work it out. The number of locations in A7 is: A x 16 + 7
= 10 x 16 + 7 = 167.

The total number of address locations which have been programmed into the EPROM is:
5 x 256 + 167
= 1447.

We emphasise this aspect of hex numbering as it is often glossed over. Values such as 5A7 do not give us any indication of the value they represent.

To program 1447 address locations would take the best part of an afternoon as it is important to check and double-check the data at each address before running a programme. It only takes one fault in the program to upset its running with the result that hours of work will be ruined.

So, 5A7 is quite a large number and it nearly fills the EPROM. In fact there are only about 600 locations left.

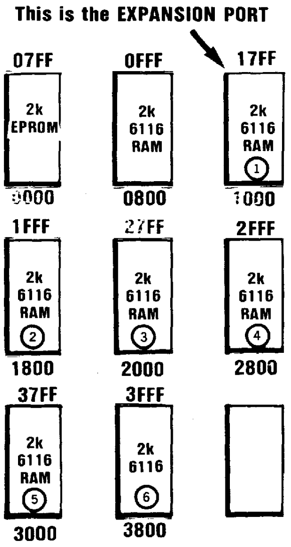
The main reason for locating the beginning of the user-available section at 0800 will become obvious in a moment.

The second and most important reason for having the first user-available location at 0800 is the value it represents.

The EPROM is a 2k byte IC and this means it is capable of storing 2048 addresses. Each address will accept a number as high as 11111111 in binary, which is 255. We casually say it is a 2k EPROM but it is actually a 2k48 EPROM.

The value 2048 is equal to 0000 to ________. Let's work it out.

Divide 2048 by 256 and you will obtain the number of "groups" of locations. This comes to 8. So, 2048 is equal to 0000 to 0800 Or more accurately 0000 to 007FF. Every 0800 in hex represents 2048 locations. The next 2k starts at 0800 to 1000 or more accurately 0800 to 0FFF.



The Memory Expansion Board, to be presented in a future article will contain 6 RAM chips. These are labelled ①-⑥ on the diagram and accessed as shown.

The hexadecimal start and finish for each "2k" of RAM on the TEC-1 is shown in the diagram above. The Z80 will access 64k of memory (65,536 bytes) or 32 chips such as 6116 or the N-MOS version 58725 by Mitsubishi. On the TEC-1, the two top address lines have not been decoded and thus the TEC will only address 16k of memory.

Between 05A7 and 07FF, the EPROM has been left blank for additional routines which will appear as MON 1A and MON 1B etc.

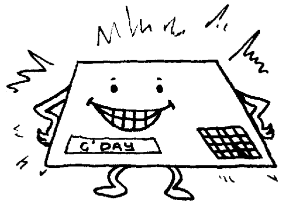
It is always wise to leave empty pockets between one program and the next in case a program needs to be extended. These empty locations can be filled at a later date with the aid of an EPROM BURNER.

Other locations in the EPROM have also been left blank. The most important of these is the "first hundred bytes". Within this section are 8 one-byte subroutine call locations such as 00, 08, 10, 18 where we can place a brief program (up to 7-8 bytes long). We can write a jump or call instruction and maybe a return instruction so that the main program needs ONLY a single byte instruction. (Normally a CALL requires 3 bytes).

You have already keyed a number of instructions and you will be starting to remember what they stand for. **3E,** for instance, means *"Load accumulator A with the following byte".* The accumulator is one of the registers in the Z80 and there is nothing magic or complex about it.

It is merely a set of 8 flip flops which can be set HIGH or LOW to reflect the number which has been entered.

The reason why register A is so often used in programmes lies in its special feature. It is the accumulator register and this means all logic operations (such as AND, OR ADDITION and SUBTRACTION) will be performed using it.



THE ALL PURPOSE NIFTY TEC-1...

Experiment 7 combines a number of interesting features such as CALL and JUMP and we will explain what these do.

The program looks fairly simple, but this very deceptive. If you were required to program all the information to produce a running letter program, it would be like being asked to buy a bottle of lemonade but firstly manufacture and print the dollar note required to buy the bottle of fizzy. Obviously this would be an enormous task so we use AIDS to make the task easier.

In our case we use John Hardy's letter writing routine and his running or shift routine and put them together to get our sentences.

We will go through the program in the same way as carried out by the Z80.

By pressing the reset button, +, +, the computer will start at location 802. The small amount of operating brains inside the Z80 will instruct it to look at location 802 in the 6116 RAM. It will find the instruction 3E and this will tell it to load the accumulator with 00. This is one way of removing the initial 'rubbish' in a register or accumulator A.

The next instruction 32 means load the contents of accumulator A (which is 00) into the address which follows, which is 0800. This means we have removed any rubbish data and loaded it with 00.

At address location 807, 'A' will be loaded with 09. The previous value, 00 will be written over but it has already done its job of being loaded into location 0800.

At location 809, the Z80 is told to load the value 09 into address 801.

All we have done to date is simply load the first available address with 00 and the second with 09. We could have done this manually but the routine calls for the contents of 800 and 801 to be altered between two different values: 0900 and 0A00 and so the computer has to do the job.

The next instruction is a CALL instruction. The Z80 is asked to look at the address 1B0 in the EPROM. What it will find there is quite considerable.

Firstly it loads register pair DE with the contents of 800 and 801. At these locations it finds the address 0900. Next it takes the first byte at 0900 and plays the note corresponding to this value. This requires another table comprising byte-pairs. One of which is the pitch value and the other time-value. It continues using each byte at 0900 until either a **1F** or **1E** is recognised. With IF the computer goes to line **80F** in the main program and carries out a similar procedure for the Letter Table.

After the letters have scrolled across the screen, the computer returns to the main program (location 817) and this instructs the Z 80 to jump to line 802.

WHAT YOU HAVE LEARNT FROM EXPERIMENTS 1-7:

1. The first available address is 0800.
2.The dots on the display indicate when the address OR the data can be changed.
3.Incrementing the display means to increase the address value.
4.Decrementing the display means to decrease the address location.
5. The TEC-1 is programmed in MACHINE CODE.
6. The key pad is a HEX PAD as it contains the numbers 0-9 and letters A-F.
7. The values 00 to FF represent the numbers ZERO to 255:

HERE IS THE COMPLETE TABLE:

OUR COMPLETE HEX TABLE

0	00								
1	01	51	33	101	65	151	97	201	C9
2	02	52	34	102	66	152	98	202	CA
3	03	53	35	103	67	153	99	203	CB
4	04	54	36	104	68	154	9A	204	CC
5	05	55	37	105	69	155	9B	205	CD
6	06	56	38	106	6A	156	9C	206	CE
7	07	57	39	107	6B	157	9D	207	CF
8	08	58	3A	108	6C	158	9E	208	D0
9	09	59	3B	109	6D	159	9F	209	D1
10	0A	60	3C	110	6E	160	A0	210	D2
11	0B	61	3D	111	6F	161	A1	211	D3
12	0C	62	3E	112	70	162	A2	212	D4
13	0D	63	3F	113	71	163	A3	213	D5
14	0E	64	40	114	72	164	A4	214	D6
15	0F	65	41	115	73	165	A5	215	D7
16	10	66	42	116	74	166	A6	216	D8
17	11	67	43	117	75	167	A7	217	D9
18	12	68	44	118	76	168	A8	218	DA
19	13	69	45	119	77	169	A9	219	DB
20	14	70	46	120	78	170	AA	220	DC
21	15	71	47	121	79	171	AB	221	DD
22	16	72	48	122	7A	172	AC	222	DE
23	17	73	49	123	7B	173	AD	223	DF
24	18	74	4A	124	7C	174	AE	224	E0
25	19	75	4B	125	7D	175	AF	225	E1
26	1A	76	4C	126	7E	176	B0	226	E2
27	1B	77	4D	127	7F	177	B1	227	E3
28	1C	78	4E	128	80	178	B2	228	E4
29	1D	79	4F	129	81	179	B3	229	E5
30	1E	80	50	130	82	180	B4	230	E6
31	1F	81	51	131	83	181	B5	231	E7
32	20	82	52	132	84	182	B6	232	E8
33	21	83	53	133	85	183	B7	233	E9
34	22	84	54	134	86	184	B8	234	EA
35	23	85	55	135	87	185	B9	235	EB
36	24	86	56	136	88	186	BA	236	EC
37	25	87	57	137	89	187	BB	237	ED
38	26	88	58	138	8A	188	BC	238	EE
39	27	89	59	139	8B	189	BD	239	EF
40	28	90	5A	140	8C	190	BE	240	F0
41	29	91	5B	141	8D	191	BF	241	F1
42	2A	92	5C	142	8E	192	C0	242	F2
43	2B	93	5D	143	8F	193	C1	243	F3
44	2C	94	5E	144	90	194	C2	244	F4
45	2D	95	5F	145	91	195	C3	245	F5
46	2E	96	60	146	92	196	C4	246	F6
47	2F	97	61	147	93	197	C5	247	F7
48	30	98	62	148	94	198	C6	248	F8
49	31	99	63	149	95	199	C7	249	F9
50	32	100	64	150	96	200	C8	250	FA
								251	FB
								252	FC
								253	FD
								254	FE
								255	FF

MOVING ON. . . .

What is the next thing you would like to do?
How about turn on one segment of the display?

Key in this program:

```
RESET
3E  01
D3  01
3E  01
D3  02
76
RESET, GO.
```

This is what you have done:
Reset 3E + 1 + D3 + 1 + 3E + 1 + D3 + 2 + 76, Reset, GO.

The top LED in the first display lights up. We say the first display as it is the lowest priority digit.

In English, this is what you have done: Load register A with the value 1. Output this to port 1. Load register A with the value 1 and output it to port 2. Halt.

This is how the program is written:

```
LD A,01      800   3E 01
OUT (1),A    802   D3 01
LD A,01      804   3E 01
OUT (2),A    806   D3 02
HALT         808   76
```

We can read the value of each location by pressing RESET and then stepping through the program by pressing: + + + + + + + + + + + + +

We can alter the position of the LED which is to be lit, by altering the value of locations 801 and 805.

The whole program does not have to be re-typed. Any location can be altered as follows:

Either press Reset and + or – – – – to get address 801.
Change 801 to 08
Press RESET, GO.

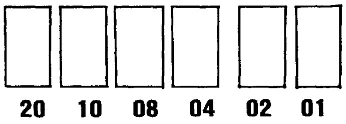
Note the LED has moved to the 4th display.

Try the values: 02, 10, 04, & 20.

You have accessed each display. Return to the first display by changing the value at 801 to: __ __
Increment to 805 and change the value 01 to: 02, 04, 08, 10, 20, 40 and finally 80.

Are you impressed? You have accessed each of the segments including the decimal point. You have become master of the display. With a little more instruction you can illuminate more than one LED in the display. But first let's see what you have learnt.

The displays are accessed as follows:



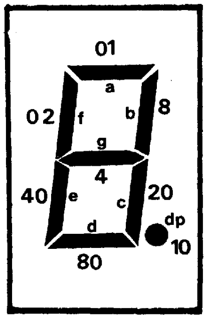
The value of each display is twice the previous and they increase from RIGHT to LEFT.

Port 1 is the cathode port. A 'HIGH' or 'ONE' in the appropriate bit 5 - 0 activates cathode 5 or 4 or 3 or 2 or 1 or 0.

This is how it works: The numbers 20, 10, 8, 4, 2, 1, are converted to binary and the computer sees them as:

```
0 0 1 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1
```

The last line in the table is easy to read. It is 1. The second lowest line is 2, then the next line has the value 4, then 8. The first and second lines are a little more difficult to explain because they are not 16 and 32. That's the binary number but we are interested in the value we have to punch into the hex keyboard. The answer will be covered in a moment. For the present, we will look into the concept of converting binary numbers into HEX numbers.



Firstly we will give you the answers. Each segment on a display has the following values. The top LED (segment A) is lit with a value 1. The decimal point needs a value 10, the centre LED (segment G) needs the value 4 and so on.

What do you think the following program will produce?

```
3E 01
D3 01
3E 20
D3 02
76
```

Try it. Start at 0800. Enter the program, press reset, GO.

We can illuminate more than one segment at a time and more than one display at a time by changing two locations as shown in the following examples:

Try each of these examples and see a pattern of addition appearing.

At location 805, insert:

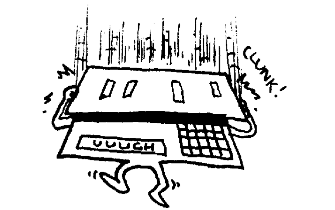
(a) 3
(b) 9
(c) 15
(d) 34
(e) 62
(f) A
(g) D
(h) F

The letters A, D and F will be quite a surprise. They also produce a reading on the display and it is obvious they have a value. Their values will be covered in the section CONVERTING BINARY TO HEX and HEX TO BINARY.

Back to the display.

More than one display can be illuminated at a time by inserting a different value at location 801. Keep say "62" at location 805 and insert the following at 801:

(a) 9
(b) 27

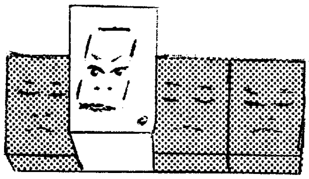


... WITH IN/OUT BOARD....

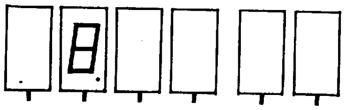
(c) 31
(d) 10
(e) A
(f) 3B
(g) 2F

This is as far as we can go with blind experimenting. We must cover some of the background theory on hex numbers to understand what we are doing.

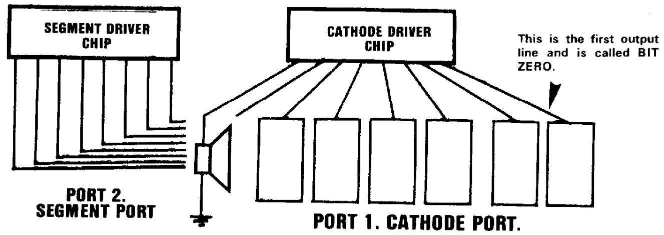
In experiment (d) above, we turned on the fifth display thus:



To do this we must switch the appropriate display transistor ON. (Don't worry about the segment drivers at this stage).



In the TEC-1 the displays are connected to the 8212's as follows:



The display we wish to illuminate is in the fifth output line and its binary number is: 00010000. The keyboard is in hex so to convert this to a hex number we have to break it into two groups of four:

0001 0000

This represents 1 This represents 0

The answer is 10.

This is not called ten. It is called one, zero or one, oh. Here are some more examples:

1. To iluminate the first, third, and fifth digits, this is the procedure:

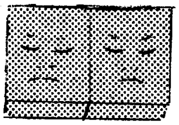
The high lines from the 8212 will be:

00101010

break this into 2 groups of four:

0010 1010

This is equal to 2 This is equal to A



WAKING UP THE FIFTH DISPLAY

The answer is 2A.

OK, you don't understand how we get A. Look at the table on P 71 of issue 10. The hex number for 1010 is A.

Try these: Write the hex number for:

(a) 1100 =
(b) 1001 =
(c) 1101 =
(d) 1111 =

2. To illuminate the first, second and third displays, the HIGHs must appear in the following places:

00111000

0011 1000

3 8

Answer: 38 (in hex)

3. To illuminate all the displays:

00111111

0011 1111

3 F

4. To bip the speaker:

10000000

8 0

5. To click the speaker and turn on displays: one, two, three and four:

10111100

1011 1100

B C

Answer: BC

6. To access the vacant cathode:

01000000

0100 0000

4 0

Answer: 40. You will notice NOTHING!

Example 3 above shows that the max hex value to illuminate ALL THE DISPLAYS is 3F. Remember this.

This is the program we are using:

```
3E 01     Max 3F
D3 01
3E 01     Max FF
D3 02
76
```

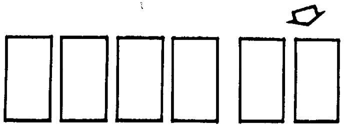
This value determines which display(s) are lit and is called CATHODE ACCESSING.

This value determines the segments which are lit and is called SEGMENT ACCESSING.

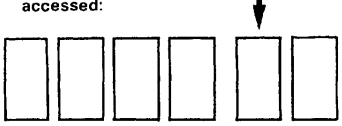
The program has two variables: lines 1 and 3. For line 1, the maximum value is **3F** (this is 64 different possibilities) and line 3 has a maximum of **FF** (this is a maximum of 256 possibilities). These are independent variables and either can be changed at any time to any value in the range as specified above. The result is thousands of different combinations.

Here are some of the possibilities and how they are obtained:

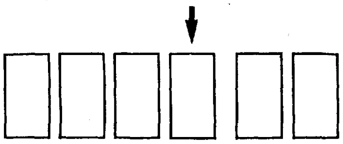
If line 1 has the value 1, we will be accessing this display: (line 3 can be any value from 00 to FF).



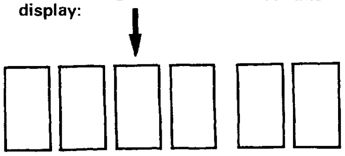
If we program **3E 02** into the first line, the following display will be accessed:



The value **3E 04** will access this display:



The value **3E 08** will access this display:

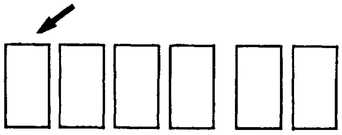


Can you see why?

The computer converts 08 to binary and it becomes 1000.
The "one" or "HIGH" corresponds to the display in the diagram above

3E 20

corresponds to this display:



Note: 20 is the Hex number and is obtained by separating 20 into 2 0.

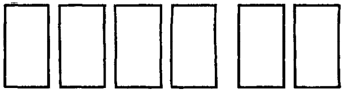
```
   2 | 0
0010 | 0000
```

This gives 0010 0000 as the binary equivalent and shows the HIGH is bit 5 and this will illuminate the first digit on the display. (Note:The first output line is bit 0).

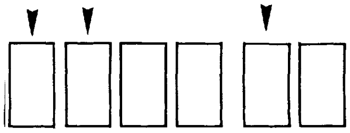
[The actual combination of segments which will be illuminated will depend on line 3 of the program and this will be explained in a moment.]

More than one display can be turned on at the same time by adding CATHODE ACCESS values. Thus 02 plus 04 will access:

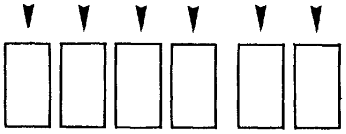
3E 06



3E 32 will access:



3E 3F will access:

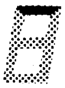


SEGMENT ACCESS

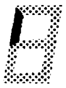
Line 3 of the program determines the letter, number or pattern which will appear in the display(s). We will take the simple case of the lowest priority display being accessed (line 1 will be 3E 01) and we will produce some interesting patterns, numbers and letters in (or on) the display.

Here are some of the results of changing the data in line 3:

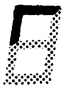
3E 01



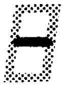
3E 02



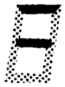
3E 03



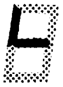
3E 04



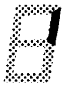
3E 05



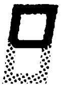
3E 06



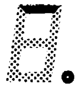
3E 08



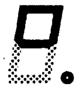
3E 0F



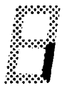
3E 11



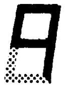
3E 1F



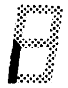
3E 20



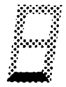
3E 2F

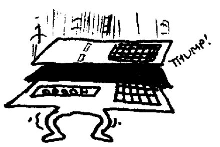


3E 40

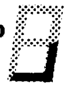


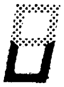
3E 80

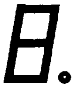




.... 8X8 MATRIX DISPLAY BOARD...

3E A0 

3E E0 

3E FF 

Here is an equally interesting program to automatically increment the SEGMENT ACCESS value. The result will be to produce every combination possible on the display. Watch the results carefully and see if you can predict the next segment to appear.

At address 0800, type the following program:

(And the DELAY ROUTINE at 0A00)

```
LD A,01        800   3E 01
OUT (1),A      802   D3 01
LD B,00        804   06 00
LD A,B         806   78
OUT (2),A      807   D3 02
INC B          80A   04
CALL 0A00      80B   CD 00 0A
JP 0806        80E   C3 06 08

 0A00

               A00   11 FF FF
               A03   1B
 This is the   A04   7B
    DELAY      A05   B2
   ROUTINE     A06   C2 03 0A
               A09   c9
```

By reducing the DELAY TIME, the display will cycle at a faster rate. Try the value 06, A0, C0, 10 and 02 for the value A02 and determine which value is the most suitable.

SUMMARY

This is the program you have been entering into the TEC-1 to illuminate the segments. The two variables are located at 801 and 805.

```
800  3E  04
802  D3  01
804  3E  10
806  D3  02
808  76
```

The data at address 801 can range from 01 to 3F and it determines the combination of digits which will be illuminated.

The data at address 805 can range from 01 to FF and it determines the combination of segments which are illuminated.

Expt 8: ILLUMINATING ONE SEGMENT

Aim: To illuminate one segment on the display .

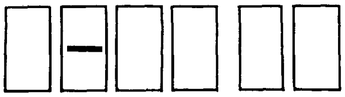
Theory: To master the display you must be able to access (locate) any segment.
The TEC-1 display has 6 digits and each has 8 light emitting diodes. This produces 6x8 = 48 locations.

The aim of this experiment is to illuminate one of these segments.

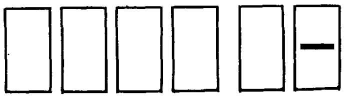
Key the following:

Reset, 3E + 10 + D3 + 01 + 3E + 04 + D3 + 02 + 76 Reset Go.

The display will show one centre segment in the 5th display thus:

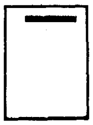


Now key in this program:
Reset 3E + 01 + D3 + 01 + 3E + 04 + D3 + 02 + 76 Reset, Go.
The result is:

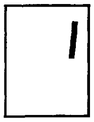


The only difference between the two programs is the byte at 801.
You are now going to make a discovery yourself:
This is how to do it:

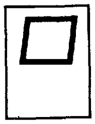
Press Reset.
Press + to 805. Change the data (now showing 04) to 01.
Press Reset, Go.
The segment 'a' illuminates thus:



Press RESET. Press + + to get address location 805.
Change data to 08.
Press RESET, GO.
Only segment b illuminates thus:

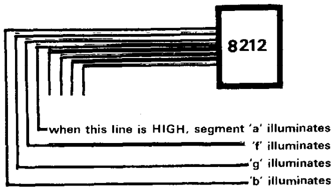


Press RESET. Press + + to 805.
Change data to OF.
Press RESET, GO.
Four segments illuminate thus:



Let's see how this comes about.

The seven-segment display is labelled a-g and the decimal point h. Each segment is illuminated via a binary number sent from the Z80 to the 8212 latches.



So far, so good.

You can see we have activated the 4 lowest value lines and this has produced a rectangle made up of segments a, b, f and g.

Now look at how we programmed a, b, f and g.
The numbers we used were 1, 2, 4, and 8.
Can you see the connection?
It's binary.

The computer converts our keyboard number to a binary number thus:

a = 1
b = 10
c = 100
d = 1000

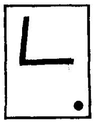
The next segment in the series is 10000, which is 16 in binary terms.

So, let's see what happens when we put 16 into the program at 801.

Key this sequence:

RESET 3E + 16 + D3 + 02 + 3E + 10 + D3 + 01 + 76 RESET GO.

Has something gone wrong? We get this result:

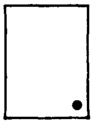


The one fact we have omitted is the TEC-1 is programmed in hexadecimal numbers. This means 16 should be typed as 10 (one, oh or one zero) since the computer counts: 1,2, 3,4,5,6,7,8,9,A,B,C,D,E,F,10. Thus 10 is equal to 10000 in binary.

Put 10 into the program thus:

RESET 3E + 10 + D3 + 02 + 3E + 10 + D3 + 01 + 76. RESET GO.

You will find the decimal point will illuminate thus:



Now add the value of segments a, b, f and g to the value of the decimal point:

This is done by adding 1 + 08 + 02 + 04 + 10 to get 1F.

Place this value in the program:

RESET*3E + 1F + D3 + 02 + 3E + 10 + D3 + 01 + 76* RESET GO.

Success.

We have just added five Hex numbers.

Let us illuminate the next segment in the series. It will have the binary value 100000.
This is equivalent to 32 in binary, but the binary value is not important. It is the hexadecimal value we are interested in. What is 100000 in Hex?
Break the value into 10 0000 so that the four last numbers form a group. The value 10 is 2 in binary and 0000 is zero. Thus the hex value is 2 0 = 20.

Place 20 in the program. Segment 'c' will illuminate.

To combine segment c with a, b, f, and g, we add their Hex values together: *20 + 01 + 08 + 02 + 04 = 2F.*

Insert 2F into the program.
Press RESET, GO.

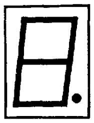
The result is a figure **NINE!**

You can now see that each number and letter in the display is produced by a set of HIGHs on the appropriate lines.

To access the next segment in the series, we must place a HIGH on the 7th line: 0 1 0 0 0 0 0 0. This is 40 in Hex terms and will result in segment 'e' iluminating. When the 8th line is HIGH, segment 'd' will be illuminated.

To illuminate ALL the segments on the display (including the decimal point) we must add the following values: *1 + 2 + 4 + 8 + 10 + 20 + 40 + 80. This is eqial to FF*
(1 + 2 + 4 + 8 + = F) (10 + 20 + 40 + 80 = F0)

Program FF into the sequence at location 801. The result is:



●●●●●●●●●●●●●●●●●●●●●●●●●●●

If you programmed our SEGMENT ACCESS routine on P 20 and watched the display carefully, you will have been amazed at the number of recognisable letters and numbers which can be created on a simple 7-segment display.

Our requirement at this stage is to be able to produce some, if not all, of the characters using the information we have learnt in experiment 8.

These are the facts we need:

1. Segment values:
a = 01
b = 08
c = 20
d = 80
e = 40
f = 02
g = 04
h = 10

2. The value of each segment is added in hex form when we need to illuminate two or more segments.

Answers to questions in col 3.

1. *2 = Cd, 3 = Ad, 4 = 2E, 5 = A7, 6 = E 7, 7 = 29, 8 = EF, 9 = 2F.*

2. *A = 6F, b = E6, C = C3, d = EC, E = C7, F = 47.*

Example: To produce the number **ONE** on a display we must acces segments b and c. This requires the value 28 (in HEX) to be inserted into the following program at location 805:

```
800  3E 01
802  D3 01
804  3E 28
806  D3 02
808  76
```

QUESTION: What would result if **28** were placed at location 801 instead of 805?

Answer: The first and third displays would illuminate with segments "a".

This means you must take care to present the data to the correct output port. In fact every program must be ABSOLUTELY CORRECT as the computer will not be able to correct any of your mistakes.

PROBLEMS:

1. What Hex value must be inserted into the program above to produce the following numbers:

1 = 28
2 = Cd
3 =
4 =
5 =
6 =
7 =
8 =
9 =

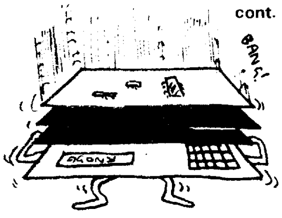
2. Work out the value to illuminate these letters:

A =
b =
C =
d =
E =
F =

3. Draw the result of entering these Hex numbers into the program:

(a) 32 =
(b) 14 =
(c) 49 =
(d) 88 =
(e) A4 =
(f) C3 =
(g) DD =
(h) F1 =
(i) FE =

cont. P.26. . .

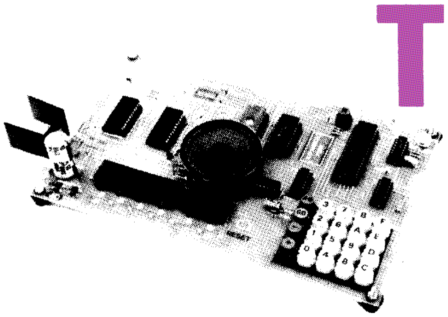


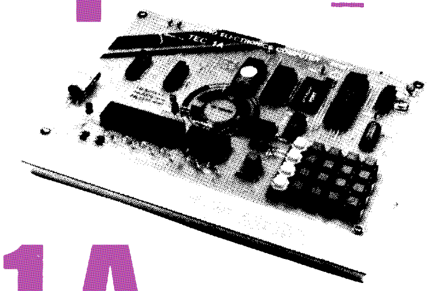
. . AND SPEECH SYNTHESIZER. . . . .

TALKING ELECTRONICS COMPUTER

TEC-1 & TEC-1A

PART III





The TEC-1A is an update of the TEC-1. For all programming, both are the same. Only TEC-1A kits are available from the kit suppliers.

FEATURES IN THIS ISSUE:

- ★ Programs for the 7-segment display.
- ★ Programs for SOUND and TONES
- ★ Programs for the 8x8
- ★ RAM STACK Project
- ★ Interface for PRINTER: PLOTTER
  - - Text mode
  - - Graphics mode
  - - Computer Graphics.

Parts: $74.00
PC board: $21.00
Case: $19.00
Post: $5.00 MAX.
'Add-ons' as required . . .

With over 1,000 TEC's in the field, we have had a lot of feedback on the success of this project.

Most of them worked first go and this was very encouraging, considering the complexity of the unit.

Five schools bought class sets and this included a University in New Zealand.

More schools and TAFE colleges will be including them in their microprocessor courses and we will see more of this type of demonstrator in the near future.

It's undeniable that the Z80 is one of the most brilliantly designed microprocessors and it is unfortunate that so little has been written about it.

This is most probably due to the designers and manufacturers of the chip. Until recently, the entire production has enjoyed advanced orders. They were extensively used in controlling applications, computers and intelligent games.

In fact the original manufacturer, Zilog, could not keep up with supplies for its own orders and franshised SGS to make the shortfall. With the down-turn of sales, we have seen some of the chips come on the market and in the initial stages, nobody knew what to do with them.

A few Z80 programming books came on the market but were generally hard to follow and severely lacking in information. Most of them didn't explain how the programming went together. This is why we have designed this series.

In the first two parts we covered some experiments to show how the display was accessed and how to create simple movement and sound effects.

In this part we are continuing along the same lines with slightly more advanced material and introduce a couple of 'add-ons' for those who are advancing faster than the midstream.

Try them all, in the order presented and you will see the concept of MACHINE CODE programming fall into place.

TEC-1A

After the first run of 1,000 boards we found the 8212 display driver chips were fairly difficult to get. So we decided to update the board and include a few small improvements at the same time. Both the TEC-1 and TEC-1A operate with the same software however the TEC-1A has 3 small changes.

The regulator is mounted under the PC board so that it cannot be bent over and broken off, the 2,200uf electrolytic has been changed to 1,000uf and the output latches have been changed to 74LS273 (or 74LS 374 or 74LS377). In all other respects, the boards are identical.

TEC CASE

Either board can be mounted on a **RETEX BOX,** size:RA-2 as shown on P.3 of issue 11. From that small photo we sold hundreds of cases, proving that the magazine is read from cover to cover.

These boxes not only neaten your project but strengthen the board near the keys. There is sufficient room inside the case to fit a power transformer however we suggest using an external plug pack connected to the computer via a plug and socket on the side of the case.

If you want a TEC-CASE, they are available from us for $19.50 including post and packing.

THE ADVENTURES OF Mr WRAGG:

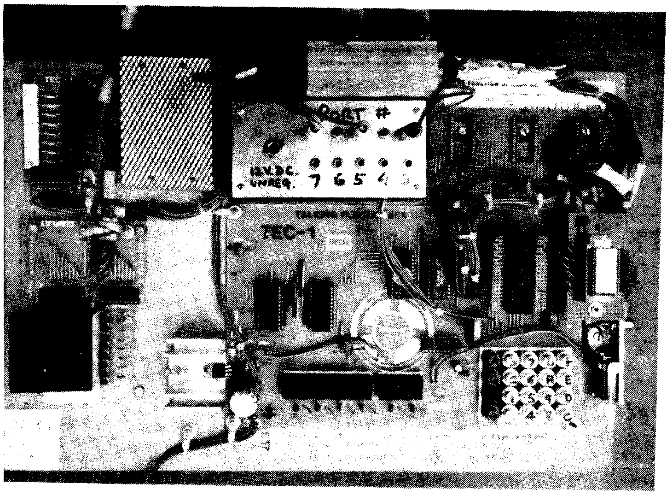
Some constructors have used their own method of mounting the PC board and others have strengthened the weaker parts of the design such as the PC board near the keys, the 7805 regulator mounting, the expansion socket etc.with a variety of ideas. But none have done more than a teacher from Brighton High School. Mr Wragg has made his computer STUDENT-PROOF. He has strengthened every lead and plug by using clips, clamps, thick hook-up wire and fasteners where ever possible to reduce the possibility of damage.

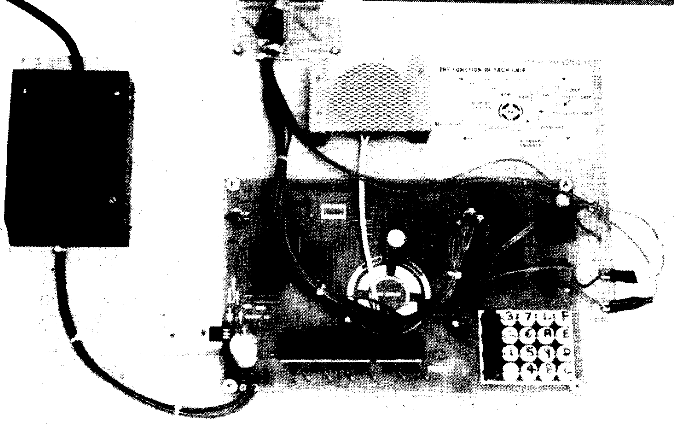
The results are shown in the photos. It looks like a before-and-after case.

The lower photo shows the TEC-1, screwed to a base-board, with the 8x8 matrix connected to the expansion socket - before he got the bug. The upper photo shows how things grew. It shows only 3 add-ons, out of the 8 he has designed and built.

Inspired by the TEC, he now holds early morning classes for electronics enthusiasts.

He covers both the practical and programming aspects of computers and some of his teaching aids will be presented in the next issue.





The beginning stages of Mr Wragg's TEC. He now has a whole batch of support devices including a Velocity-Acceleration recording interface & a programmed ROM.

We will be passing on may of these developments in future issues.

Programming on the 7-Segment Displays:

We continue from P.29 of issue 11 with more programs on the TEC display.

For this, you will need the TEC-1 or TEC-1A. No 'add-ons' are used at this stage. The displays on the PC provide the readout.

The next program, **"Back and Forth"** runs the 'g' segment across the 6 displays and back again without shifting to the 7th and 8th outputs. Thus the blank output and speaker are not activated.

The number of 'shifts' is determined by the value loaded into register C and 1, 2, 3, 4, 5, or 6 displays can be activated.

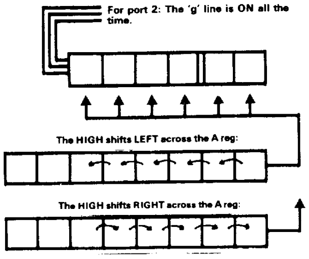
"BACK AND FORTH"

```
LD A,04 ('g' segment)       800   3E 04
OUT (2),A (seg. port)       802   D3 02
LD C,05                     804   0E 05
LD A,01 1st display         806   3E 01
OUT (1),A (cath. port)      808   D3 01
LD B,A                      80A   47
CALL DELAY                  80B   CD 00 09
LD B,A                      80E   78
RLC A                       80F   CB 07
DEC C                       811   0D
JP NZ LOOP                  812   C2 08 08
LD C,06                     815   0E 06
OUT (1),A                   817   D3 01
LD B,A                      819   47
CALL DELAY                  81A   CD 00 09
LD A,B                      81D   78
RRC A                       81E   CB 0F
DEC C                       820   0D
JP NZ LOOP 2                821   C2 17 08
JP START                    824   C3 00 08
```

Delay at 0900:

```
11 FF 07
1B
7B
B2
C2 03 09
C9
```

For Port 1, the cathode of the first display is activated, then the 2nd display, then the third display, etc.


For port 2: The 'g' line is ON all the time.
The HIGH shifts LEFT across the A reg:
The HIGH shifts RIGHT across the A reg:

The diagram shows the shifting of the HIGH across the accumulator (the A register). This is output to port 1 and will turn on one transistor at a time. These transistors feed the common cathode lines of the displays.

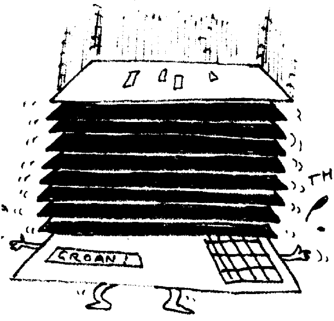
Keep this program in the computer for the next experiment.

QUESTIONS:

1. At 805, why do we insert 05 into register C? Try the value 06 and see what happens. (slow the clock rate to see the effect.)

2. At 805, insert the value 03. What happens?

3. At 805, insert the value 01. Can you see what is happening on the screen and in register C. (The bit is being shifted through the register via the speaker and is appearing on the left hand side of the display.)



AND OUTPUT LATCHES & DRIVERS.

Another interesting TEC-1 repair came in this week. The constructor had purchased a set of components from us in 'short-form'. He had bought some of the items himself and the remainder he purchased from us - things like the EPROM, Z80 and display drivers.

But what was interesting, he had made the printed circuit board himself. Not being satisfied with photocopying the layout on the back of the magazine, he reproduced the entire artwork using tape and stick-on lands. It was copied so exactly that it took us a few minutes to realize the situation. And when it finally dawned, it was quite a shock! It was like looking at a forgery! The work entailed in its creation must have been enormous. And now we had to repair it.

Normally there is not a single firm under the sun which would entertain a repair which had not been made with components and PC board as supplied by the organisation. You can imagine the assortment of repairs which would be sent in.

But because we have had so much success with repairing the computer, we accepted it and went to work.

Theoretically, every track had to be inspected because the board was a 'one-off'.

After quite a few minutes, we noticed two tracks did not connect to the appropriate places. When we joined these with short lengths of tinned copper wire, all the chips received power, but still the computer failed to work.

Then we noticed the clock chip was a Fairchild 4049 and this was promptly replaced.

The display immediately lit up and on going through the keyboard, we found keys 4 and 7 did not give the correct display reading.

Back to more inspection. More detective-work located a track on the common line which should not have been there. When this was cut, the computer worked perfectly.

Here's the lesson:

If you are going to reproduce artwork, you must check, re-check and then triple check the layout. The only effective way to do this is to make the artwork identical in size and design so that the two layouts can be placed together and held up to the light. Looking from both sides, you will be able to detect any discrepancy in the wiring.

When the board is etched and drilled, go over the entire board with a multimeter and check each of the lines for continuity. Mark each line as you check it so that none are missed.

Only this way can you be sure every line goes to its correct destination.

I know it is fun to produce everything yourself, but for some items the result is not economical. Making PC boards is one of the most time-consuming and costly endevours. After its all finished, it will probably cost nearly as much as a bought one and won't have an overlay or solder mask.

If you have made your own PC board and it doesn't work, please don't send it in for repair. We have met the challenge and don't need it again. It will involve too much of our time and delays the production of the next issue of the magazine.

"ALL THE VALUES"

This program produces all the combinations for the 7-segment display. This will include many unusual effects as well as all the known letters and numbers.

It is basically an extension to the **BACK and FORTH** program with an additional listing at **0A00** and a change at **824** and **801.**

Fill in this section:

```
3E 00
D3 02
0E 05
3E 01
D3 01
47
CD 00 09
78
CB 07
0D
C2 08 08
0E 06
D3 01
47
CD 00 09
78
CB 0F
0D
C2 17 08
C3 02 0A
```

At 0A00:

```
LD L,01    A00   2E 01
INC L      A02   2C
LD A,L     A03   7D
JP 0802    A04   C3 02 08
```

Delay at 0900:

```
11 FF 07
1B
7B
B2
C2 03 09
C9
```

Push RESET, GO:

Write the ASSEMBLY CODE for the program above and also the address listings, starting at 0800.

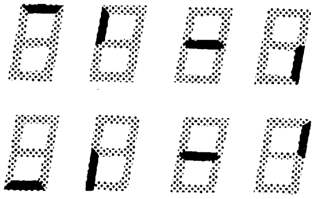
MOVEMENT AROUND A SINGLE 7-SEGMENT DISPLAY:

The following program produces movement around a single 7-segment display which can be increased in speed to produce a novel effect.

```
LD A,20      800   3E 20
OUT (1),A    802   D3 01
LD A,01      804   3E 01
CALL 0900    806   CD 00 09
LD A,02      809   3E 02
CALL 0900    80B   CD 00 09
LD A,04      80E   3E 04
CALL 0900    810   CD 00 09
LD A,20      813   3E 20
CALL 0900    815   CD 00 09
LD A,80      818   3E 80
CALL 0900    81A   CD 00 09
LD A,40      81D   3E 40
CALL 0900    81F   CD 00 09
LD A,04      822   3E 04
CALL 0900    824   CD 00 09
LD A,08      827   3E 08
CALL 0900    829   CD 00 09
JP 0800      82C   C3 00 08
```

at 0900:

```
OUT (2),A    900    D3 02
             902    11 FF 07
             905    1B
             906    7B
             907    B2
             908    C2 05 09
             90B    C9
```



This program has not been efficiently written. It contains a repetition of **LD A'** and **'CALL'** instructions. It should have a **BYTE TABLE.** This will be shown in a later program.

However it does contain one byte-saving feature. The statement **OUT (2),A** is used at each stage and has been placed in the **CALL ROUTINE.** This saves 14 bytes of program.

1. Replace **(07)** at **904** with **00** and watch the screen.

2. Replace **(FF)** at **903** with **0F**.

3. At address **801** replace the data with **01, 04,** and then **10.** What happens to the display?

4. Replace data at **801** with **05, 0F, 2D.** What happens to the display in each of these cases?

THE DELAY ROUTINE

We have used a delay routine in many of the programs we have investigated.

We have also seen how it can be adjusted from a few milliseconds to a few seconds in length.

For this time-delay to occur, many thousands of clock cycles must be involved in its execution. In fact, up to one million or more cycles can be involved.

Let us look at how this comes about and how the delay operates.

The delay program we will be investigating is:

```
1.   11 FF 07
2.   1B
3.   7B
4.   B2
5.   C2
```

The meaning of each line is:

1. **LD DE07FF** FF is loaded into E and 07 into D.
2. **DEC DE** Register E is decremented by one. If an **underflow** occurs, register D is decremented.
3. **LD A,E** Register E is loaded into the accumulator.
4. **OR D** The D register is OR-ed with the accumulator.
5. **JP NZ** to: The program jumps to line 2 if the result is not zero.

The number of cycles to perform each operation is as follows:

LD DE	**11 FF 07**	10 cycles
DEC DE	**1B**	6 cycles
LD A,E	**7B**	7 cycles
OR D	**B2**	7 cycles
JP NZ Line 2	**C2 03**	10 cycles

One loop consists of:

1B	- 6
7B	- 7
B2	- 7
C2	- 10

total: 30 cycles.

To produce a delay of 256 loops, the instruction is:

11 FF 00
FF is loaded into the E register and 00 into the D register.

E is decremented by one on each loop of the program and when it gets to 00, the result of OR-ing the accumulator (which will contain the value of the D register - 00) will be zero and the microprocessor will jump out of the delay routine and back to the main program.

Total clock time for the 256 loops is:
256 x 30
= 7680 cycles.

If the D register is loaded with FF the delay time will be:
    7680 x 256
    = 1,966,000 cycles.
This is about 2 million cycles!

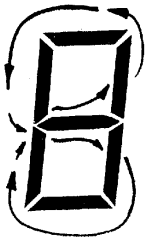
When deciding upon a delay of suitable length, try various values in this location:

11 FF 07

This location has only a small effect on the delay time and can be considered to have a 'trimming effect'.

FIGURE 8's ACROSS THE SCREEN:

The next program introduces a table of values which the program 'looks up' during the execution of each cycle. These are called **'data bytes'** or bytes of data, which are used one at a time.



The LED turn-on sequence around the display.

```
LD C,20        800    0E 20
LD A,C         802    79
OUT (1),A      803    D3 01
CALL 0900      805    CD 00 09
RRC C          808    CB 09
BIT 7, C       80A    CB 79
JP Z LOOP      80C    CA 02 08
JP 800         80F    C3 00 08
```

```
LD HL,0B00     900    21 00 0B
LD B,9         903    06 09
LD A(HL)       905    7E
OUT (2)A       906    D3 02
CALL DELAY     908    CD 00 0A
INC HL         90B    23
DEC B          90C    05
JP NZ 905      90D    C2 05 09
RET            90F    C9
```

At 0A00

```
               A00    11 FF 07
               A03    1B
               A04    7B
               A05    B2
               A06    C2 03 0A
               A09    C9
```

at 0B00:

```
B00   01
B01   02
B02   04
B03   20
B04   80
B05   40
B06   04
B07   08
B08   01
```

On the first pass, the program places 0B00 into a register pair such that the 0B goes into the H register (meaning the High order byte register) and 00 in to the L register (Low order byte register).

At 905 the contents of the byte at 0B00 will be loaded into the Accumulator because this is the address specified by the HL instruction.

On each subsequent pass, the HL register pair in incremented by ONE. Since the value 0B00 is contained in this pair, the result will be to add 1 to 00 to get 01, 02 etc. Thus the program will look up 0B01, 0B02 etc and finds the relevant byte of data.

The second interesting part of this program is the counting of the **DATA BYTE** table. The computer must know how many data bytes are to be accessed.

Thus it is given an intial value of **09** at **904** and decremented the value by one on each pass. When the result is zero, the program jumps to **0800** and starts again.

The segment can be made to travel across the screen in the opposite direction by changing 3 values:

The starting value at 801 must be changed to 01, RRC must be changed to RLC **(CB 01)** and bit 6 must be tested for zero **(CB 71)**

The changes are:

```
800  0E 01
808  CB 01
80A  CB 71
```

CONTROL VIA THE KEYBOARD

Movement on the screen can be controlled by the keyboard by introducing a **HALT** or wait function. This causes the program to halt and wait for an input via the interrupt line.

When a key is pressed, the non-maskable interrupt line is activated and allows the Z80 to accept data from the keyboard encoder via the data bus.

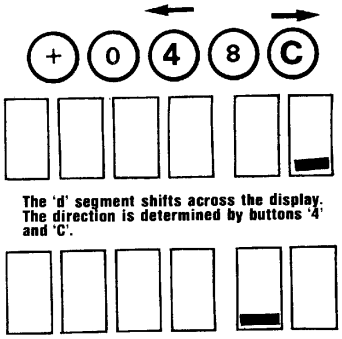
The data is loaded into the accumulator and compared with a value in the program. If the two values are the same, the output is zero and as determined by the next instruction, the program advances.

This program moves a LED across the bottom row.

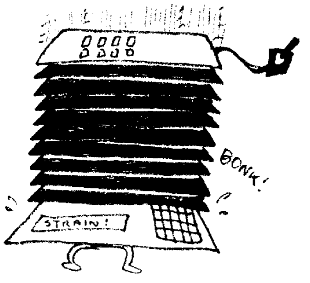
Key '4' shifts the LED left and 'C' shifts it right.

The direction of shift is determined by **RLC B** and **RRC B.** Each press of a button moves the LED one place. No delay routine is required in this program.

```
LD A,04        800    3E 04
OUT (2),A      802    D3 02
LD B,A         804    47
LD A,B         805    78
OUT (1),A      806    D3 01
HALT           808    76
LD A,01        809    ED 57
CP 04          80B    FE 04
JP NZ 815      80D    C2 15 08
RLC B          810    CB 00
JP 805         812    C3 05 08
CP 0C          815    FE 0C
JP NZ 808      817    C2 08 08
RRC B          81A    CB 08
JP 805         81C    C3 05  08
```



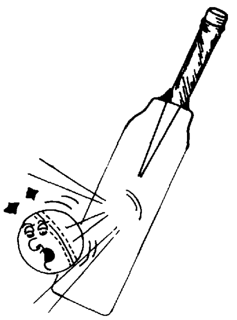
The 'd' segment shifts across the display. The direction is determined by buttons '4' and 'C'.



.. VIDEO DISPLAY UNIT....

CREATING A BAT

This program produces a **2-segment** bat capable of travelling across the lower segment of the display. The '+' key moves the bat to the left and the 'C' key moves it to the right.



```
LD A,80        800    3E 80
OUT (2),A      802    D3 02
LD B,03        804    06 03
LD A,B         806    78
OUT (1),A      807    D3 01
HALT           809    76
LD A,I         80A    ED 57
CP 10          80C    FE 10
JP NZ 0816     80E    C2 16 08
RLC B          811    CB 00
JP 806         813    C3 06 08
CP C           816    FE 0C
JP NZ 809      818    C2 09 08
RRC B          81B    CB 08
JP 806         81D    C3 06 08
```

WRITING A PROGRAM

This is a written exercise requiring YOU to write a program. Our aim will be to write a BAT program exactly like the previous program and you can refer to it if a problem arises.

For each line, the MACHINE-CODE value should be obtained from the Z80 CODE SHEET on the back page of issue 11. It should then be placed in the space provided.

1. Load the accumulator with the value 80. Answer:____________
2. Output the contents of the accumulator to port 2. ____________
3. Load register B with the value 3: ____________
4. Load register B into the accumulator: ____________
5. Output the accumulator to port 1: ____________
6. Halt the program: ____________

The Z80 is now waiting for an interrupt.

7. Load the index register into the accumulator: ____________
8. Compare the accumulator with the value 10: ____________
9. Jump to 'COMPARE C' (below) if the answer to line 8 is NOT zero: ____________
10. Rotate register B LEFT CIRCULAR: ____________
11. Jump to the address which states: Load B into A (above): ____________
12. Compare the accumulator with the value 'C': ____________
13. Jump to HALT (above) if NOT zero: ____________
14. Rotate register C Right Circular: ____________
15. Jump to: Load register B into A (above): ____________

Complete the following listing by adding the values you have obtained from the statements above:

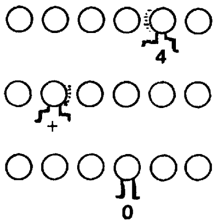
```
800
802
804
806
808
80A
80B
80D
80F
812
814
815
817
81A
81D
81F
822
824
825
827
82A
```

AUTO MOVEMENT & HALT

S Riley,
Guildford, 2161.

The following program detects 3 keys. The '+' key shifts the LED left, the '0' key stops the LED and key '4' shifts it right.

The speed of travel across the display is controlled by the DELAY ROUTINE.



```
LD A,01        800    3E 01
OUT (2),A      802    D3 02
LD A,01        804    3E 01
OUT (1),A      806    D3 01
LD B,01        808    06 01
HALT           80A    76
LD A,I         80B    ED 57
CP '+'         80D    FE 10
JP NZ 081D     80F    C2 1D 08
RLC B          812    CB 00
LD A,B         814    78
OUT (1),A      815    D3 01
CALL DELAY     817    CD 00 0C
JP 80B         81A    C3 0B 08
CP 04          81D    FE 04
JP NZ 80D      81F    C2 0D 08
RRC B          822    CB 08
LD A,B         824    78
OUT (1),A      825    D3 01
CALL DELAY     827    CD 00 0C
JP 808         82A    C3 0B 08

   at 0C00:           11 FF 0A
                      1B
                      7B
                      B2
                      C2 03 0C
                      C9
```

So far we have turned on one segment or LED at a time in the display or more than one segment or LED within the same digit. But not 2 LEDs in different displays, in different positions.

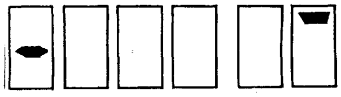
This seems impossible but by using a clever pulsing technique we can alternately access one then the other to produce the effect of both being on at the same time.

In this program we will alternately access segment 'g' in the first display and segment 'a' in the 6th display to give the appearance that they are both on at the same time.

SWITCHING 2 PIXELS INDEPENDENTLY:

Run the following program and observe the effect:

```
LD A,20        800    3E 20
OUT (1),A      802    D3 01
LD A,01        804    3E 04
OUT (2),A      806    D3 02
CALL DELAY     808    CD 00 0B
JP 0A00        80B    C3 00 0A
```



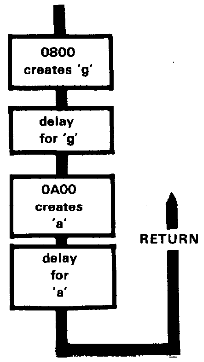
```
LD A,01        A00    3E 01
OUT (1),A      A02    D3 01
LD A,01        A04    3E 01
OUT (2),A      A06    D3 02
CALL DELAY     A08    CD 00 0B
JP 0800        A0B    C3 00 08
```

at 0B00:

```
                    11 FF 00
                    1B
                    7B
                    B2
                    C2 03 0B
                    C9
```

Turn the speed control up and the effect is two different LEDs being lit at the same time. Turn the speed control down and the alternating effect becomes more noticeable.

The flow diagram for this is:



Insert **05**into the delay routine at **0B02** and watch the display. The alternating effect is more obvious. This is the basis for all the letters and writing on the display. Each digit is being turned on and off very quickly.

PROBLEMS:

Turn the speed control up and keep the delay routine short for the following problems:

1. Change values in the program to turn on segment 'd' in the first display and 'a' in the sixth display.

2. Create the figure 1 in the first display and '0' in the last display. Which locations in the program must be altered to achieve this?

TO CONTROL 2 PIXELS. One with movement.

This program produces two pixels. One is fixed and the other moves up and down.

In this experiment, the main program is at **0A00** and it calls the delay at **0B00** and a short routine at**0800**. When the program has been entered, push, RESET, ADdress, 0A00, GO, GO to execute the program. The main task with this experiment will be to rewrite the main program so that it appears at 0800. This will involve changing a number of machine code values to suit the new location.

```
LD C,BB (or any value)  0A00  0E BB
LD A,01                 0A02  3E 01
OUT (1),A               0A04  D3 01
LD A,01                 0A06  3E 01
OUT (2),A               0A08  D3 02
CALL DELAY              0A0A  CD 00 0B
CALL 0800               0A0D  CD 00 08
DEC C                   0A10  0D
JP NZ 0A02              0A11  C2 02 0A
LD C,BB                 0A14  0E BB
LD A,01                 0A16  3E 01
OUT (1),A               0A18  D3 01
LD A,04                 0A1A  3E 04
OUT (2),A               0A1C  D3 02
CALL DELAY              0A1E  CD 00 0B
CALL 0800               0A21  CD 00 08
DEC C                   0A24  0D
JP NZ 0A16              0A25  C2 16 0A
LD C,BB                 0A28  0E BB
LD A,01                 0A2A  3E 01
OUT (1),A               0A2C  D3 01
LD A,80                 0A2E  3E 80
OUT (2),A               0A30  D3 02
CALL DELAY              0A32  CD 00 0B
CALL 0800               0A35  CD 00 08
DEC C                   0A38  0D
JP NZ 0A2A              0A39  C2 2A 0A
JP 0A00                 0A3C  C3 00 0A
```

Delay at 0B00:

```
                              11 FF 00
                              1B
                              7B
                              B2
                              C2 03 0B
                              C9
```

```
LD A,20       800   3E 20
OUT (1),A     802   D3 01
LD A,01       804   3E 80
OUT (2),A     806   D3 02
CALL DELAY    808   CD 00 0B
RETURN        80B   C9
```

Problem:

1. Rewrite the MAIN PROGRAM to start at 0800:

```
LD C,BB       800
LD A,01       802
OUT (1),A     804
LD A,01       806
OUT (2),A     808
CALL DELAY    80A
CALL 0A00     80D
DEC C         810
JP NZ 0802    811
LD C,BB       814
LD A,01       816
OUT (1),A     818
LD A,04       81A
OUT (2),A     81C
CALL DELAY    81E
CALL 0A00     821
DEC C         824
JP NZ 0816    825
LD C,BB       828
LD A,01       82A
OUT (1),A     82C
LD A,80       82E
OUT (2),A     830
CALL DELAY    832
CALL 0A00     835
DEC C         838
JP NZ 082A    839
JP 0800       83C
```

Fill in the MACHINE CODE values:

at 0B00:
Same Machine code values for delay.

```
                        0B00
                        0B03
                        0B04
                        0B05
                        0B06
                        0B09
```

at 0A00:

```
LD A,20       A00
OUT (1),A     A02
LD A,01       A04
OUT (2),A     A06
CALL DELAY    A08
RETURN        A0B
```

Run the program by pressing RESET, GO. Does it work? (It should)

2. Insert the following data into the program you have written, to produce the name of a semiconductor:

at 0806: 3E 47

at 081A: 3E C7

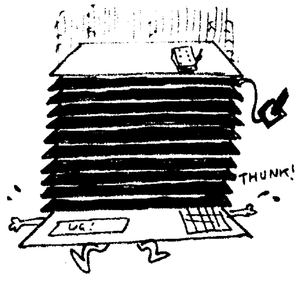
at 082E: 3E C6

What is the name of the device?

3. Create the name of another semiconductor device by inserting the following information into the program:

```
802   3E 10
806   3E 4F
816   3E 08
81A   3E EA
82A   3E 04
82E   3E C6
```

Remove the value **20** at **0A01.**
Push RESET, GO.
What is the name of the device?
Reduce the delay of **BB** in the MAIN PROGRAM (at 3 locations) to **05**.
The result will be your first readable multiplexed word.



AND AN EPROM BURNER.

JUMPS AND CALL INSTRUCTIONS

JUMP and CALL instructions are called **BRANCH INSTRUCTIONS.**

They cause the program to branch to another location in memory and execute the instruction contained at that location.

The 6 instructions we will investigate are:

JP Address	**C3 XX XX**
JP NZ Address	**C2 XX XX**
JR dis	**18 XX**
JR NZ dis	**20 XX**
CALL Address	**CD XX XX**
CALL NZ Address	**C4 XX XX**

The meaning of each instruction is as follows:

JP Address. This is an unconditional instruction . It means **Jump: Address.** The program will jump to a new address as determined by the next two bytes XX XX.

JP NZ Address. This means **Jump, non-zero: address.** The program will only jump to a new address if the result of the previous instruction is **NOT** zero. (If the result is zero, the program will neglect this 3-byte instruction and advance to the next instruction).

JR dis. This is an unconditional statement. It means: **Jump relative: displacement.**

In simple terms a relative jump means the program will jump to an address of plus 129 bytes or minus 126 bytes of the address of the JR op-code byte.

For instance, the value **FB** will cause a jump to **1B** in the following program.

```
11 FF 07
1B
7B
B2
18 FB
C9
```

For a forward jump, **03**will cause the program to jump to **D3** in the following:

```
3E 01
18 03
3E 20
D3 01
3E 28
```

JR NZ dis. This is a conditional statement. It means **Jump relative, non-zero: displacement.** The displacement is given by a hex value such as **D7, EE, F8** for a backward jump or **07, 18, 44, 76,** for a forward jump.

When determining the displacement value, this is an easy method:

```
★★ ★★          F9 FA
★★             FB
★★ ★★          FC FD
20 XX          FE FF
★★ ★★ ★★       00 01 02
★★ ★★          03 04
★★             05
               etc.
```

can be **18** or **20**

CALL Address. This is an unconditional instruction. It means **CALL the address given by the next two bytes XX XX.**

When using this instruction, it must be the intention of the programmer to call a sub-routine and then return to the instruction which immediately follows, as this is the requirement of the microprocessor.

For this reason, the sub-routine must conclude with a return instruction **C9.** The address of the byte immediately following **CD XX XX** will be saved in the stack. At the conclusion of the sub-routine it will be **popped** off the stack, looked at, and cause the program to return to the instruction after **CD XX XX.**

CALL NZ Address. This is a conditional instruction and will only be executed if the result of the previous instruction is **NOT** zero.
All other features of this instruction as per CALL Address above.

The main differences between these three sets of jump instructions are:

A **JP** instruction causes the program to go to a sub-routine but does not call it back again.

A **JP** instruction can make the program go to any location in memory. It is not restricted to a displacement value.

A **JP** instruction cannot be re-located without changing or looking at the two-byte jump address to see if the sub-routine is still at the same address.

A **JR** can only operate within +127 and -128 bytes (approx.)

JR can be easily re-located as it relates only to relative memory. This type of instruction is ideal when large portions of a program need to be shifted.

CALL instructions are used when a sub-routine is required to be executed (such as a delay) followed by a return to the main program.

QUESTIONS

1. Write the meaning of these, in words:

(a) **JP**
(b) **JP NZ**
(c) **JR dis**
(d) **JR NZ dis**
(e) **CALL**
(f) **CALL NZ**

2. Which instruction would you use for the following:
(a) You require to go to a sub-routine and then return to the main program.
(b) You require to go to another routine if the answer to the previous line is NOT zero.
(c) You require to go to the beginning of the program.
(d) You require to go to a location about 15 bytes further down the program.
(e) You require to go to a sub-routine on the condition NON-zero, and return.
(f) You require to go to a location back 8 bytes.

3. Give one advantage of a **JUMP RELATIVE** instruction, compared to a **JUMP** instruction.

4. To produce a loop in a program, which of the following should be used: **JR dis or JR NZ dis.**

5. At the end of a program, which instruction should be used: **CALL, JR NZ, JP.**

6. What is the difference between **CALL** and **JUMP**?

Answers:

1. Jump.
Jump Non-zero.
Jump Relative displacement.
Jump relative non-zero displacement
Call.
Call Non-zero.

2. (a) **CALL**
(b) **JP NZ**
(c) **JP**
(d) JR 14
(e) **CALL NZ**
(f) **JR F6**

3. The program can be transferred to another location without affecting the **JUMP RELATIVE** instruction.

4. It must have a non-zero condition.

5. It must be a **JUMP** instruction with no other conditions.

6. **CALL** transfers the program to another location and requires that it be returned to the next instruction after the sub-routine has been performed. **JP** transfers the program to another location without any return requirement.

USING JR's

To show how we can substitute a **JR** instruction for a **JUMP** instruction, we will consider a simple program containing a delay routine.

We will choose the program: **RUNNING SEGMENT 'a' ACROSS THE SCREEN.** This can be found on P. 26 of issue 11 and is repeated here:

Type this into the computer and **RUN.**

at 0800:

```
LD A,01         800   3E 01
OUT (2),A       802   D3 02
LD B,01         804   06 01
LD A,B          806   78
OUT (1),A       807   D3 01
CALL DELAY      809   CD 00 0A
RLC B           80C   CB 00
JP LOOP         80E   C3 06 08

           0A00     11 FF FF
                    1B
                    7B
                    B2
                    C2 03 0A
                    C9
```

1. We will change the instruction at **80E** to **JR 806** . Change the address values to **18 F6.**

Place this into the program and **RUN.** Is any difference observed? (There should be no difference).

2. The delay routine at **0A00** can also be changed to include a **JR** instruction.

at **A06:**
change **C2 03 0A** to **20 FB 00**

Run the program and note the result. No difference should be detected. Both instruction perform the same in this case.

3. The **DELAY PROGRAM** can be placed immediately below the main program so that a **JR** instruction can be used at **809** and also at the end of the delay.

at **809:** insert **JR 820.**

Start the delay routine at **0820.**

At the end of the delay routine, insert:
JR 80C.

The displacement values will have to be worked out by you. Follow through the steps as shown and write the complete program. Use the TEC as a counter to work out the displacement values (by pushing +,+,+,+,+, etc.)

Do not look at the answer at the bottom of the next column until you have finished.

Next issue we will give a **JR table** and explain how it is used.

QUICK DRAW

In this final exercise we will change a number of **JUMP** instructions to **JR** instructions. See the **QUICK DRAW** program on P. 13 of issue 11.

This is how to change the program:

1. Copy all the assembly code, replacing **JP** with **JR.**

2. Copy the machine code listing, remembering that the 3-byte **JP** instructions will become 2-byte **JR** instructions. At this stage do not insert the displacement values - this will be the final job.

3. Insert the displacement values for each of the **JR** instructions.

4. Fill in the memory locations, starting at **0800.**

Push RESET, GO and play the Quick Draw game. Does everything work correctly? It should.

We have learnt the major advantage of a **JR** instruction. It enables a program to be transferred to another location without having to alter any of the data.

See the effectiveness of this. Move the whole Quick Draw game to **0900** or **0A00** making sure you wipe the program at **0800** before starting at the new location.

To start the game, push RESET, GO, GO. Is it a success?

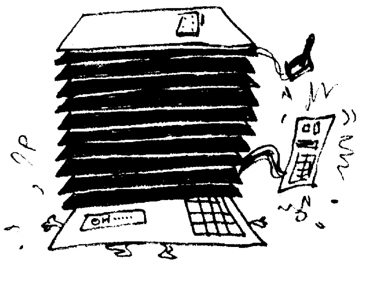
This is what your program should look like:

QUICK DRAW

```
        LD A,00        800   3E 00
        OUT (1),A      802   D3 01
START   LD DE,00       804   11 00 00
DELAY   DEC DE         807   1B
        LD A,D         808   7A
        OR E           809   B3
        JR NZ Delay    80A   20 FB
        LD A,E3        80C   3E E3
        OUT (2),A      80E   D3 02
        LD A,08        810   3E 08
        OUT (1),A      812   D3 01
LOOP 1  HALT           814   76
        AND OF         815   E6 0F
        CP 0C          817   Fe 0C
        JR Z,Right     819   28 05
        OR A           81B   B7
        JR Z,Left      81C   28 06
        JR Loop 1      81E   18 F4
RIGHT   LD A,01        820   3E 01
        JR Finish      822   18 02
        LD A,20        824   3E 20
        OUT (1),A      826   D3 01
LEFT    LD A,20        828   3E 28
FINISH  OUT (2),A      82A   D3 02
        HALT           82C   76
        JR Start       82D   18 D1
```

Your final program will look like this:

```
LD A,01        800     3E 01
OUT (2),A      802     D3 02
LD B,01        804     06 01
LD A,B         806     78
OUT (1),A      807     D3 01
JR 820         809     18 14
RLC B          80B     CB 00
JR 806         80D     18 F7

               820     11 FF FF
               823     1B
               824     7B
               825     B2
 JR NZ 823     826     20 FB
 JR 80B        828     18 E0
```



...AND REMOTE CONTROL....

TONES & TUNES

Here is a selection of tones and tunes for the computer. These have been submitted by readers and are presented in various formats to get you acquainted with the different ways of presenting a program.

OSCILLATOR

by Peter Alekseievs

```
800   3E  LD A,80
801   80
802   D3  OUT (01),A
803   01
804   3E  LD A,00
805   00
806   D3  OUT (01),A
807   01
808   C3  JP 800
809   00
80A   08
```

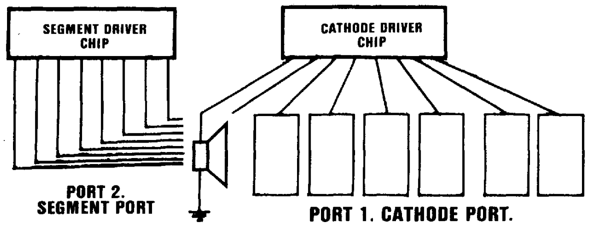
The principle of operation of this program can be seen in the diagram. We are accessing the speaker via port 1 and this is the 8th line of the driver chip. Thus the value 80 is inserted in the program.

We load a HIGH into this line for a number of clock cycles and then a LOW. This produces a CLICK which sounds like an oscillator when the speed control is increased.

TOCCATA

-by Stephen Clarke, 2774.

```
800: 00 00 3E 00 32 00 08 3E 09 32 01 08 CD B0 01 CD
810: B0 01 3E 21 32 00 08 CD B0 01 3E 62 32 00 08 CD
820: B0 01 C3 02 08

900: 0A 0F 11 12 0F 11 12 14 11 12 14 16 12 14 16 17
910: 14 16 12 14 11 12 0F 11 0E 0F 0A 0B 08 0A 0A 00
920: 1F 00 17 17 17 17 16 16 16 16 14 14 14 14 12 12
930: 12 12 11 11 11 11 0F 0F 0F 0F 0E 0E 0E 0E 0E 0E
940: 0E 0E 0B 0B 0B 0B 0A 0A 0A 0A 08 08 08 08 06 06
950: 06 06 05 05 05 05 03 03 03 03 02 02 02 02 02 02
960: 00 1F 0A 0F 11 12 0F 11 12 14 11 12 14 16 12 14
970: 16 17 14 16 12 14 11 12 0F 11 0E 0F 0A 0B 08 0A
980: 08 06 05 03 03 03 03 03 03 03 03 03 00 00 00 00
990: 00 00 00 00 00 00 1F
```



PHONE RING

By Cris Cogdon.

This program generates a ring similar to that of a new phone. It would make an ideal trick if you have one of these phones!

```
START   CALL RING       800    CD 14 08
        LD HL,1000      803    21 00 10
        CALL PAUSE      806    CD 1E 08
        CALL RING       809    CD 14 08
        LD HL,8000      80C    21 00 80
        CALL PAUSE      80F    CD 1E 08
        JR START        812    18 EC

RING    LD B,10         814    06 10
XRING   PUSH BC         816    C5
        CALL 081E       817    CD 8E 01
        POP BC          81A    C1
        DJ NZ XRING     81B    10 F9
        RETURN          81D    C9

PAUSE   DEC HL          81E    2B
        LD A,H          81F    7C
        OR L            820    B5
        RET Z           821    C8
        JR PAUSE        822    18 FA
```

FREQUENCY SWEEP

by Peter Alekseievs

This program gives an effect similar to a phaser gun. By changing the value of the second byte, different effects can be generated.

This program can be placed anywhere in memory as it consists entirely of **JR** instructions.

```
LD H,FF       26 FF
LD B,H        44
LD A,00       3E 00
OUT (1),A     D3 01
LD A,80       3E 80
OUT (1),A     D3 01
LD A,B        78
DEC A         3D
JR NZ FD      20 FD
DJNZ F2       10 F2
LD B,00       06 00
LD A,00       3E 00
OUT (1),A     D3 01
LD A,80       3E 80
OUT (1),A     D3 01
INC B         04
LD A,B        78
DEC A         3D
JR NZ FD      20 FD
LD A,H        7C
SUB B         90
JR NZ EF      20 EF
JP DA         18 DA
```

THE STRIPPER

```
800: 00 00 3E 00 32 00 08 3E 09 32 01 08 CD B0 01 C3
810: 02 08

900: 01 01 03 03 03 06 06 06 06 06 06 0A 08 08 06 06
910: 06 02 02 02 02 02 02 01 01 03 03 03 06 06 06 06
920: 06 06 0A 0D 0D 0C 0C 0C 0B 0B 0B 0B 0B 0B 0B 0A
930: 0A 01 01 01 01 0A 0A 01 01 01 01 0A 0A 09 0A 0A
940: 0A 0B 0B 0A 0B 0B 0C 0D 0D 0C 0D 0D 0D 0E 0E 0D
950: 0E 0E 0E 0F 0F 0E 0F 0F 11 11 11 0F 11 11 12 12
960: 12 12 12 12 12 00 00 00 00 00 00 00 00 00 00 1F
```

TEC CLOCK

This program will allow the TEC to be used as a CLOCK. The display is used as the readout and the time can be set as shown opposite.

This is a 24 hour clock and its accuracy depends on the setting of the SPEED CONTROL. In a future issue we will present a crystal oscillator to take the place of the 4049 to turn the TEC into an accurate time-piece.

To set CLOCK:

at 989: insert seconds
at 98A: insert minutes.
at 98B: insert hours.

Example: 7:45:32

989: 32 98A: 45 98B: 07

```
START   LD IY, Clock Buffer        900   FD 21 89 09   Load pointer to clock counting buffer
        LD B,2                     904   06 02         load number of 60's to be tested
        LD A,(IY +0)               906   FD 7E 00      Read first clock buffer value
        ADD A,01                   909   C6 01         add 1 to the value
        DAA                        90B   27            decimal adjust the accumulator
        CP 60                      90C   FE 60         TEST A=60  sec/min
        JR NZ,DISP                 90E   20 13         GOTO 'DSP' if not equal
        XOR A                      910   AF            ZERO the accumulator
        LD (IY+0),A                911   FD 77 00      Store A in clock buffer
        INC IY                     914   FD 23         Advance pointer
        DJNZ EE                    916   10 EE         complete LOOP if B is not zero
        LD A,(IY +0)               918   FD 7E 00      Read hours value
        ADD A,01                   91B   C6 01         Increment hours value
        DAA                        91D   27            Decimal adjust the accumulator
        CP 24H                     91E   FE 24         TEST hours =24
        JR NZ,DISP                 920   20 01         If not GOTO 'DSP'
        XOR A                      922   AF            ZERO A
DISP    LD (IY + 0),A              923   FD 77 00      Store hours in clock buffer
        LD B,03                    926   06 03         Load number of bytes to be converted
        LD HL,DISP BUF +6          928   21 92 09      Load pointer to display buffer
        LD IX,CLK BUF              92B   DD 21 89 09   Load pointed to clock buffer
LOOP 1  LD A,(IX + 0)              92F   DD 7E 00      Read CLOCK BUFFER value
        INC IX                     932   DD 23         Advance pointer by 1
        PUSH BC                    934   C5            Save BC contents
        PUSH AF                    935   F5            Save contents of A
        AND 0F                     936   E6 0F         Get least significant 4 bits
        LD B,A                     938   47            Transfer A to B
        CALL LOOK                  939   CD 73 09      Get pattern for B
        POP AF                     93C   F1            Restore AF
        SRL A                      93D   CB 3F         Shift A one place to the right
        SRL A                      93F   CB 3F
        SRL A                      941   CB 3F
        SRL A                      943   CB 3F
        LD B,A                     945   47            Load A into B
        CALL LOOK                  946   CD 73 09      Get bit pattern for B
        POP BC                     949   C1            Restore BC
        DJNZ LOOP                  94A   10 E3         Complete LOOP if B is not zero.
        LD B,0FFH                  94C   06 FF         Load LOOP value
LOOP 2  LD IX.DISP BUF             94E   DD 21 8C 09   Load pointer to digit patterns
        PUSH BC                    952   C5            Save BC contents
        LD B,07                    953   06 07         Load number of digits to be displayed
        LD C,40H                   955   0E 40         Load bit pattern for display cathodes
        LD A,(IX +0)               957   DD 7E 00      Read display pattern
        OUT (2),A                  95A   D3 02         Output pattern to port 2
        LD A,C                     95C   79            Load C into A
        OUT (1),A                  95D   D3 01         Output cathode pattern to port 1
        SRL C                      95F   CB 39         Move cathode bit one place for MUX effect
        XOR A                      961   AF            Clear A
        LD E,10H                   962   1E 10         Load TIME DELAY value
        DEC E                      964   1D            Decrement E
        JR NZ FD                   965   20 FD         LOOP if not equal to zero
        OUT (1),A                  967   D3 01         Turn off anode bits
        INC IX                     969   DD 23         Advance to next pattern
        DJNZ loop 2                96B   10 EA         LOOP if not zero
        POP BC                     96D   C1            Restore BC
        DJNZ LOOP 2                96E   10 DE         LOOP if all digits not displayed
        JP START                   970   C3 00 09      Jump to START

LOOK    LD DE, DISP                973   11 7F 09      Load DE with display pattern
        PUSH AF                    976   F5            Save AF
        LD A,E                     977   7B            Load E into A
        ADD A,B                    978   80            Calculate pattern address
        LD E,A                     979   5F            Load A into E
        LD A,(DE)                  97A   1A            Read pattern
        LD (HL),A                  97B   77            Store pattern in display buffer
        DEC HL                     97C   2B            Decrement HL
        POP AF                     97D   F1            Restore AF
        RETURN                     97E   C9            End of sub-routine

        DISP PATTERN: EB, 28, CD,  97F
          AD, 2E, A7, E7, 29, EF, AF.
        CLOCK BUFFER               989
        DISP BUFFER                98C
```

SPIROID ALIENS

·by M Allison, 3095

This is quite a long program and shows the length of listing required to achieve a degree of realism. The game uses all of page **0800** and portions of **0900, 0A00, 0B00** and **0D00.**

The main program is at **0800** with calls at the other pages.

The game consists of unusual-shaped aliens passing across the display. Each game consists of 16 passes and you must shoot down the arrivals by pressing buttons 1, 2 or 3. To win, you must shoot down at least 11.

In the initial stages of the game, you must acquaint yourself with the connection betweeen the spiroid shapes and buttons 1, 2, 3. After this you will be ready to launch an attack.

Here's the listing:

```
Reserved for
message.          800  00 08
Blank             802  00
LD HL,903         803  21 03 09
LD A,80           806  3E 80
LD (HL),A         808  77
INC HL            809  23
LD A,00           80A  3E 00
LD C,A            80C  4F
LD (HL),A         80D  77
LD HL,0911        80E  21 11 09
LD A,20           811  3E 20
LD (HL),A         813  77
INC HL            814  23
LD A,00           815  3E 00
LD (HL),A         817  77
LD HL, 0849       818  21 49 08
LD A,01           81B  3E 01
LD (HL),A         81D  77
LD B,10           81E  06 10
LD D,00           820  16 00
LD IY,0865        822  FD 21 65 08
CALL 0D00         826  CD 00 0D
LD A,(0865)       829  3A 65 08
CP C              82C  B9
JR Z, 0826        82D  28 F7
LD HL,0849        82F  21 49 08
                  832  00
CP 01             833  FE 01
JR Z,083F         835  28 08
CP 02             837  FE 02
JR Z,0843         839  28 08
LD A,61           83B  3E 61
JR, 0845          83D  18 06
LD A,0F           83F  3E 0F
JR 0845           841  18 02
LD A,26           843  3E 26
LD (084D),A       845  32 4D 08
LD A( )           848  3E ( )
OUT A,(01)        84A  D3 01
LD A,(SYMBOL)     84C  3E ( )
OUT A,(02)        84E  D3 02
CALL 0900         850  CD 00 09
CALL 090E         853  CD 0E 09
XOR,A             856  AF
IN A(00)          857  DB 00
```

```
LD E,A            859  5F
LD A,72           85A  3E 72
CP E              85C  BB
JR Z,0861         85D  20 02
LD E,00           85F  1E 00
LD A,E            861  7B
AND 03            862  E6 03
CP (CODE)         864  FE ( )
JR Z,0871         866  28 09
SLA (HL)          868  CB 26
LD A,40           86A  3E 40
CP (HL)           86C  BE
JR Z,0878         86D  28 09
JR 0848           86F  18 D7
LD C,A            871  4F
INC D             872  14
CALL 0B00         873  CD 00 0B
JR 087D           876  18 05
LD C,00           878  0E 00
CALL 0A00         87A  CD 00 0A
LD A,01           87D  3E 01
LD (HL),A         87F  77
DJNZ 0822         880  10 A2
LD HL,0807        882  21 07 08
LD A,0B           885  3E 0B
CP D              887  BA
JR C,08A5         888  38 1B
LD A,(HL)         88A  7E
CP F0             88B  FE F0
JR Z,0842         88D  28 03
ADD A,10          88F  C6 10
LD (HL),A         891  77
LD HL,0800        892  21 00 08
LD A,F8           895  3E F8
LD (HL),A         897  77
PUSH HL           898  E5
CALL 01B0         899  CD B0 01
POP HL            89C  E1
LD A,E8           89D  3E E8
LD (HL),A         89F  77
CALL 0270         8A0  CD 70 02
JR 08BE           8A3  18 19
LD A,(HL)         8A5  7E
CP 10             8A6  FE 10
JR Z,08AD         8A8  28 03
SUB 10            8AA  D6 10
LD (HL),A         8Ac  77
LD HL,0800        8AD  21 00 08
LD A,DE           8B0  3E DE
LD (HL),A         8B2  77
PUSH HL           8B3  E5
CALL 01B0         8B4  CD B0 01
POP HL            8B7  E1
LD A,CA           8B8  3E CA
LD (HL),A         8BA  77
CALL 0270         8BB  CD 70 02
LD A,3F           8BE  3E 3F
OUT A,(01)        8C0  D3 01
LD A,8A           8C2  3E 8A
OUT A,(02)        8C4  D3 02
HALT              8C6  76
JP 0802           8C7  C3 02 08
                  8CA  00
Messages: |       8CB  01
          |       8CC  0C
          |       8CD  09
          ↓       8CE  05
                  8CF  0D
                  8D0  12
                  8D1  00
                  8D2  04
                  8D3  05
                  8D4  12
                  8D5  13
                  8D6  11
                  8D7  0E
```

```
                  8D8  16
                  8D9  05
                  8DA  04
                  8DB  1A
                  8DC  00
                  8DD  1F
                  8DE  04
                  8DF  00
                  8E0  04
                  8E1  00
                  8E2  04
                  8E3  00
                  8E4  01
                  8E5  01
                  8E6  01
                  8E7  1F
                  8E8  00
                  8E9  05
                  8EA  0D
                  8EB  04
                  8EC  00
                  8ED  0E
                  8EE  06
                  8EF  00
                  8F0  05
                  8F1  01
                  8F2  11
                  8F3  13
                  8F4  08
                  8F5  1A
                  8F6  00
                  8F7  1F
                  8F8  01
                  8F9  1A
                  8FA  01
                  8FB  1A
                  8FC  01
                  8FD  1A
                  8FE  1F
```

```
PUSH AF           B00  F5
PUSH BC           B01  C5
PUSH DE           B02  D5
PUSH HL           B03  E5
LD HL(0903)       B04  2A 03 09
PUSH HL           B07  E5
LD HL(0911)       B08  2A 11 09
                  B0B  E5
LD HL,0912        B0C  21 12 09
LD A,00           B0F  3E 00
LD(HL),A          B11  77
DEC HL            B12  2B
LD A,20           B13  3E 20
ld (HL),A         B15  77
LD HL,0904        B16  21 04 09
LD A,00           B19  3E 00
LD(HL),A          B1B  77
DEC HL            B1C  2B
LD A,24           B1D  3E 24
LD (HL),A         B1F  77
LD HL,0B35        B20  21 35 0B
LD A,01           B23  3E 01
LD (HL),A         B25  77
EXX               B26  D9
LD DE,0904        B27  11 04 09
LD C,00           B2A  0E 00
LD HL,0B68        B2C  21 68 0B
LD B,06           B2F  06 06
LD A,01           B31  3E 01
LD (DE),A         B33  12
LD A,01           B34  3E 01
OUT (01),A        B36  D3 01
LD A,(HL)         B38  7E
OUT (02),A        B39  D3 02
```

```
CALL 0900        B3B  CD 00 09
LD A,00          B3E  3E 00
LD (DE),A        B40  12
CALL 090E        B41  CD 0E 09
INC HL           B44  23
DEC DE           B45  1B
EX DE,HL         B46  EB
DEC (HL)         B47  35
EX DE,HL         B48  EB
INC DE           B49  13
DJNZ 0B31        B4A  10 E5
EXX              B4C  D9
SLA (HL)         B4D  CB 26
EXX              B4F  D9
INC C            B50  0C
LD A,06          B51  3E 06
CP C             B53  B9
JP Z,0B5A        B54  CA 5A 0B
JP 082C          B57  C3 2C 0B
Exx              B5A  D9
POP HL           B5B  E1
LD (0911),HL     B5C  22 11 09
POP HL           B5F  E1
LD (0903),HL     B60  22 03 09
POP HL           B63  E1
POPDE            B64  D1
POPBC            B65  C1
POPAF            B66  F1
RETURN           B67  C9
                 B68  01
                 B69  09
   LOOK-UP       B6A  29
   TABLE         B6B  A9
   FOR           B6C  E9
   SPIRAL        B6D  EB


PUSH AF          A00  F5
PUSH BC          A01  C5
PUSH HL          A02  E5
LD HL(0903)      A03  2A 03 09
PUSH HL          A06  E5
LD HL(0911)      A07  2A 11 09
PUSH HL          A0A  E5
LD B,09          A0B  06 09
LD HL,0911       A0D  21 11 09
LD A,05          A10  3E 05
LD (HL),A        A12  77
INC HL           A13  23
LD A,00          A14  3E 00
LD (HL),A        A16  77
LD HL,0903       A17  21 03 09
LD A,1F          A1A  3E 1F
LD (HL),A        A1C  77
INC HL           A1D  23
LD A,00          A1E  3E 00
LD (HL),A        A20  77
DEC HL           A21  2B
CALL 090E        A22  CD 0E 09
DEC (HL)         A25  35
LD A,01          A26  3E 01
CP (HL)          A28  BE
JP Z 0A2F        A29  CA 2F 0A
JP 0A22          A2C  C3 22 0A
DJNZ 0A1A        A2F  10 E9
POP HL           A31  E1
LD (0911),HL     A32  22 11 09
POP HL           A35  E1
LD (0903),HL     A36  22 03 09
POP HL           A39  E1
POP BC           A3A  C1
POP AF           A3B  F1
RETURN           A3C  C9
```

```
LD A,R           C00  ED 5F
CALL 03B5        C02  CD B5 03
AND 03           C05  E6 03
LD E,A           C07  5F
LD A,00          C08  3E 00
CP E             C0A  BB
JR Z 0C00        C0B  28 F3
LD A,E           C0D  7B
CP C             C0E  B9
JR Z 0C00        C0F  28 EF
RETURN           C11  C9

PUSH HL          D00  E5
PUSH BC          D01  C5
LD HL,0D06       D02  21 06 0D
LD B,01          D05  06 01
LD A,R           D07  ED 5F
DJNZ, 0D07       D09  10 FC
AND 08           D0B  E6 08
PUSH HL          D0D  E5
LD LH,0D33       D0E  21 33 0D
ADD A,L          D11  85
LD L,A           D12  6F
LD E,HL          D13  5E
LD HL,0D33       D14  21 33 0D
LD B,08          D17  06 08
LD C(HL)         D19  4E
INC HL           D1A  23
LD A,(HL)        D1B  7E
DEC HL           D1C  2B
LD (HL),A        D1D  77
INC HL           D1E  23
DJNZ,0D1A        D1F  10 F9
LD (HL),C        D21  71
POP HL           D22  E1
INC (HL)         D23  34
LD A,20          D24  3E 20
CP (HL)          D26  BE
JR Z,0D2F        D27  28 06
                 D29  FD 73 00
                 D2C  C1
   LOOK-UP       D2D  E1
   TABLE         D2E  C9
   FOR           D2F  36 01
   RANDOM        D31  18 F6
   NUMBERS       D33  01
                 D34  02
                 D35  03
                 D36  01
                 D37  02
                 D38  03
                 D39  01
                 D3A  02
                 D3B  03
```

Finally, the listing at **0900** must be inserted. This listing can be found in issue 11 P 36, under the heading **ALIENS ATTACK RUN.** This will provide the sound for the game.

This completes the listing. Before pushing **RESET, GO**, it is a very good idea to go through the complete listing again and double-check each of the machine code values. The reason for this is to prevent the prograrn **SELF DESTRUCTING.** This could happen if you placed the wrong value in one of the locations which caused the computer to write over some of the contents of the program.

PUSH & POP

PUSH and **POP** are very much like **PUSH** and **PULL**. They are operations which transfer the contents of a register-pair to a holding area so that the registers can be used for other operations. This holding area is called the **STACK.**

We say register PAIR because the operations **PUSH** and **POP** require that 2 registers be specified. Thus, if the accumulator (Register A) is required to be pushed onto the stack, we combine it with the FLAGS register to get the register pair: AF.

There are a few technical complications concerning the placement of bytes onto the stack but these will not concern us at this stage. It is sufficient to say that the stack is located at the top end of the RAM, (about 8 - 10 bytes from the top)and as each new set of bytes is placed on the stack, the pile grows DOWNWARDS, towards the program we are executing.

We have already seen the effect of placing (PUSHING) more and more bytes onto the stack (issue 11, P. 12) and for this reason we must use the stack very carefully. Otherwise it will increase downwards and and crash into our program!

Basically we PUSH one pair of bytes onto the stack (from say register-pair AF) then push another pair of bytes onto the stack from say register pair HL. This will leave the accumulator and HL registers free for other operations.

If we want to get the 2 bytes of AF from the stack, we must firstly POP the two bytes from HL and then we can get the AF pair. It is a simple principle of **LAST ON, FIRST OFF.**

Pushing and popping are very handy instructions. By using a PUSH instruction at the start of a routine and a POP at the end, we can place a routine such as a delay routine, which will not affect the registers at all. This routine is said to be TRANSPARENT.

PUSHING and POPPING can take place between the stack and register pairs including the index registers. This group consists of the following: AF, BC, DE, HL,IX and IY.

It is interesting to note that the bytes are pushed onto the stack HIGH BYTE first, then LOW BYTE. They come off the stack LOW BYTE then HIGH BYTE. But because the stack is increasing DOWNWARDS, each byte placed onto the stack will have a lower address!

In the programs we have presented you can see PUSH and POP in operation. The stack is a temporary holding area and only the top pair can be accessed.

TALKING ELECTRONICS

35 Rosewarne Ave Cheltenham 3192

COMPLETE LIST OF TE KITS:

bankcard **584 2386**

( ) AC Plug Pack	9.20
( ) Black Jack	8.50
( ) " " PC Board	2.50
( ) Capacitance Meter	4.15
( ) " " PC Board	2.10
( ) Clock	19.50
( ) " " PC Board	3.30
( ) Clock Large Display	5.40
( ) Combination Lock	5.40
( ) " " PC Board	2.10
( ) Counter Module	17.90
( ) " " PC Board	3.80
( ) Seven Segment Display	6.90
( ) " " PC Board	4.50
( ) Cube Puzzle	13.60
( ) " " PC Board	4.30
( ) Designer Board (mother)	3.45
( ) Designer Board #1	4.00
( ) Designer Board #2	4.00
( ) Designer Board #3	4.00
( ) Matrix Board 24x25	2.10
( ) 3 Boards 24x25	5.75
( ) Type 200 + 600 board	4.30
( ) 5 type 200 + 640	17.25
( ) Fibre Glass 200+ 640	5.20
( ) Kit of 6 (1 of each)	18.40
( ) Digi Chaser	14.40
( ) " " PC Board	3.00
( ) Diode Tester	1.50
( ) " " PC Board	1.80
( ) Door Chime	13.00
( ) " " PC Board	2.00

PROJECTS THIS ISSUE:

( ) BIG EAR - see FM BUG.

( ) Headlight Reminder	**9.60**
( ) " " PC board	**3.30**
( ) Printer Interface	**24.80**
( ) " " PC board	**3.80**
( ) Ram Stack - per 2k:	**9.70**
( ) Touch Puzzle	**4.15**
( ) " " PC board	**2.10**

( ) Dual Power Supply (Kens)	8.00
( ) " " PC Board	3.80
( ) Dual Tracking Supply	15.60
( ) " " PC Board	3.30
( ) Egg Timer	6.20
( ) " " PC Board	2.10
( ) 8-watt Amplifier	7.20
( ) " " PC Board	2.50
( ) Experimenter Board 1-8	12.20
( ) " " PC Board	2.10
( ) Experimenter Deck 1-10	13.25
( ) " " PC Board	4.30
( ) FM Bug	5.00
( ) " " PC Board	1.80
( ) Hangman	9.20
( ) " " PC Board	3.30
( ) IC Pocket Radio	8.70
( ) " " PC Board	2.10
( ) Jiffy Box	2.20
( ) LED Dice MkII	5.75
( ) " " PC Board	3.30
( ) LED Zeppelin	5.75
( ) " " PC Board	2.10
( ) Light the LED	3.50
( ) " " PC Board	1.10
( ) Logic Designer	17.60
( ) " " PC Board	3.30
( ) Logic Probe	8.75
( ) " " PC Board	2.80
( ) Bread Board WB - 2N	13.20
( ) Lotto Selector	13.20
( ) " " PC Board	3.30
( ) Mini Frequency Counter	15.60
( ) " " PC Board	3.30
( ) Cascade, red LEDs	9.80
( ) " " PC Board	3.30
( ) Mini Freq, green LEDs	18.10
( ) Mini Mixer	2.20
( ) " " PC Board	1.70
( ) Music Colour	7.80
( ) " " PC Board	2.10

( ) Noise-A-Tron	3.30
( ) " " PC Board	2.00
( ) Phaser Gun	5.10
( ) " " PC Board	2.10
( ) Pill Timer	5.80
( ) " " PC Board	2.10
( ) Plug Pack 200mA AC	9.20
( ) Power Supply 1 amp	4.40
( ) " " PC Board	2.10
( ) Programmable Counter	8.20
( ) " " PC Board	3.60
( ) Auto Reset Section	7.20
( ) " " PC Board	2.50
( ) Complete Kit	21.00
( ) Quick Draw	1.80
( ) " " PC Board	2.10
( ) Square Wave Oscillator	2.60
( ) " " PC Board	2.10
( ) Simplicity Amp 1ch	4.40
( ) " " PC Board	2.40
( ) 2 channels	8.80
( ) " " 2 PC Boards	4.80
( ) Pre Amp Section	9.80
( ) " " PC Board	3.40
( ) Rouled	4.90
( ) " " PC Board	3.60
( ) Stage-1 Complete Pack	74.75
( ) Stereo Mini Mixer	19.60
( ) " " PC Board	3.50
( ) Stereo VU Meter	10.70
( ) " " PC Board	3.30
( ) Super Bug	6.00
( ) " " PC Board	2.10
( ) TEC-1A TE COMPUTER	74.00
( ) " " PC Board	21.00
( ) 8x8 Display	16.80
( ) " " PC Board	6.00
( ) Relay Driver Board	19.40
( ) " " PC Board	9.00
( ) Train Signals	5.10
( ) " " PC Board	2.60
( ) Throttle MkII	2.50
( ) " " PC Board	1.80
( ) Tremolo	2.00
( ) " " PC Board	1.80
( ) 500mA DC Plug Pack	14.50
( ) TEC case	19.50

( ) **Next 4 publications to TE (with PC's)**	**.15.00**
( ) **6 issue subscription to TE without PC's**	**.12.00**
( ) Project book No. 1: **MINI FREQUENCY COUNTER**	**3.95**
( ) Project book No. 2: **LOGIC DESIGNER**	**.3.95**
( ) Project book No. 3: **DUAL TRACKING SUPPLY**	**.3.95**
( ) Project book No. 4: **3-DIGIT DVM** DIGITAL VOLTMETER	**3.95**
( ) Project book No. 5: **LOGIC DESIGNER MK II**	**.3.95**
( ) **FULL SERIES** of above project books	**.19.50**

Postage on FULL SERIES: $1.50

( ) **ELECTRONICS Stage-1**	**.2.95**
( ) **ELECTRONICS Stage-2** TO BE RELEASED	**.3.30**
( ) **DIGITAL ELECTRONICS** REVEALED	**.2.90**
( ) **ELECTRONICS NOTEBOOK 1**	**.2.40**
( ) **ELECTRONICS NOTEBOOK 2** JUST OUT!	**.2.60**
( ) **ELECTRONICS FOR MODEL RAILWAYS** JUST OUT!	**.3.30**
( ) **TE COVER PROJECTS** JUST OUT	**.4.00**
( ) **CMOS DATA BOOK** JUST OUT	**.3.00**

Postage 90¢ per book

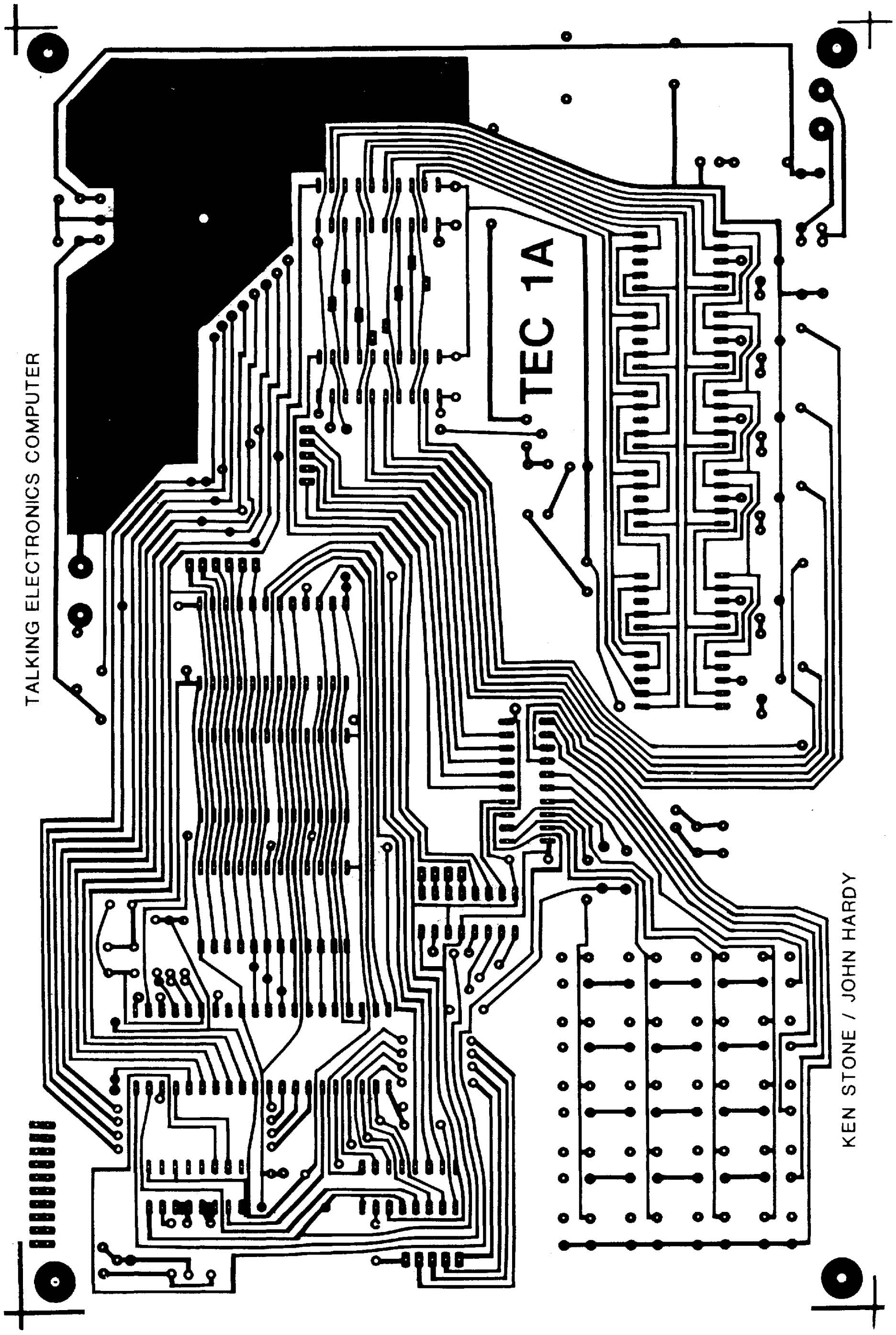
( ) **MAGAZINE BINDERS $5.90** plus $2.00 post & pack

Don't forget to order the PC board for each kit.

Pack and post: $1.50 per kit. Computer kit or PC board: $2.50 each. Small items such as PC boards are 80¢ for the first board and 50¢ for each additional board.

Maximum pack and post: **$5.00**

For orders below $10.00 please send stamps (30¢, 50¢, 80¢ $1) or cash. A $2.40 cheque costs $1.00 to process through the bank!!


TALKING ELECTRONICS COMPUTER
TEC 1A
KEN STONE / JOHN HARDY