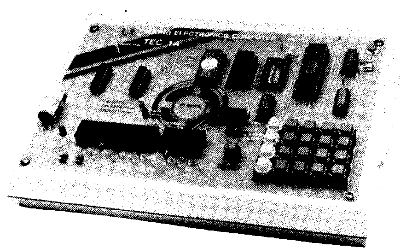


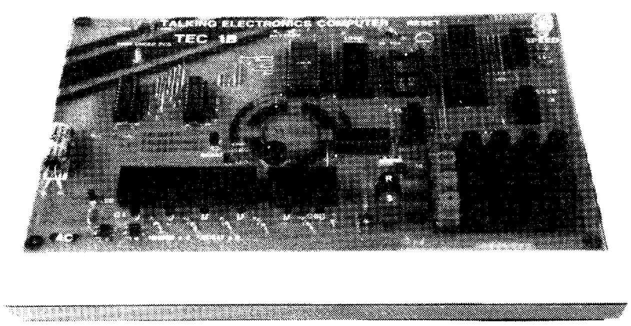
$98 COMPLETE
PLUS $6.00 POST

TALKING ELECTRONICS COMPUTER

PART IV

TEC-1A & TEC-1B

TEC 1A's can be converted to TEC 1B's by adding 1 push button, 1 47k resistor and 1 diode. Update to MON 2 and you have a SHIFT key for functions such as INSERT, DELETE etc.



TEC-1B board with SHIFT and RESET keys in foreground.

PC board: $21.00
Parts for 1B: $77.00
Case: $21.50
Post: $6.00 MAX.

FEATURES IN THIS ISSUE:
- ★ **NON-VOLATILE RAM**
- ★ **EPROM BURNER**

SEE ALSO:
TEC POWER SUPPLY on P. 23.

This is the fourth article on the TEC and introduces you to more Machine Code programming as well as two valuable add-ons.

The NON-VOLATILE RAM has been a real boon for assisting in program preparation for the MICROCOMP-1 project described in this issue.

Program can be written directly into RAM and by changing the switch, the contents will be retained for up to a year via the batteries mounted on the board.

This is the answer to ..ll those requests from constructors wanting a battery backed-up system or tape-save facility. When the TEC is turned off, the contents of memeory will be saved and thus allow you to move the TEC from one location to another.

The RAM can also be used in place of an EPROM for the purpose of getting a system up and running. When you are satisfied with the design, the program can be transferred to EPROM.

This is where our second 'add-on' comes in. We have designed an EPROM BURNER to fit on the EXPANSION PORT socket.

With all the add-ons connected to the TEC, it was soon realized that the power required was more than could be supplied from a plug pack or 2155 transformer.

This led us to design a power supply exclusively for the TEC and at the same time include all the voltage values needed for the various projects.

So far we need 5v for the electronics, 12v for the relays and 26v for the EPROM BURNER.

The TEC POWER SUPPLY is capable of delivering these and can be expanded to about 1.4 amps at 5v by paralleling two 2155's.

Don't forget, the DC current capability of a 2155 is .7amps and NOT 1 amp and this has been covered in a previous article starting on page 5 of issue 11.

As you can see, one thing leads to another and we have sufficient add-ons to turn the TEC into a powerful programming tool.

The TEC itself has changed too. From the original TEC model, we improved the layout and upgraded the output latches to modern 20 pin types and mounted the regulator under the board so that it would not be broken off.

We have now upgraded the TEC to model 1B and this has seen the inclusion of a shift key.

This shift feature allows the keyboard to have a second command for each key and opens up a world of possibilities.

Two functions which have been lacking on the TEC are INSERT and DELETE. With the addition of the shift key, you will be able to make corrections to your programs and close up gaps as well as create locations for new instructions.

Those who have already built the TEC can add a shift key in one of two ways. The lower RESET key can be converted into a SHIFT function by wiring a resistor and diode into circuit and connecting to the computer. The only problem with this is the upper RESET button. It will be difficult to access when the Video Display unit is mounted over the Z-80/EPROM area.

A better solution is to drill 4 holes near the lower RESET button and add the necessary components under the board.

The shift function is software controlled and you will need the updated MON 2 to get the shift key to work.

The MON 2 also includes a few other improvements. The most noticeable of these is the location of the STACK. You will remember the original position of the stack is very close to the top of the 6116.

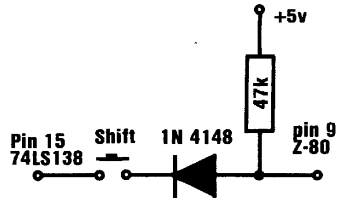
The main problem with this location is not knowing how far you can program before running the risk of hitting the stack.

The MON 2 places the stack at **08C0** and allows up to **C0** bytes to be stored. There is still a risk of crashing the computer if a stack error occurs as the stack grows down to **0000** and restarts at **FFFF** and will eventually hit the top of your program. Between C0 and **FF** is a storage area for pointers, restarts, display buffer, keyboard buffer, register save area and for interrupts.

This means programming starts at **0900** up to **0FFF** with the on-board 6116 and you don't have the problem of landing in the stack area.

You can upgrade to MON 2 by sending in your ROM and it will be re-burnt to MON 2. The cost is $3.50 plus $1.50 post.

A shift button, 47k resistor and signal diode is available in a separate kit for 80¢ and this will double the capability of your computer.



Adding Shift to TEC-1 , TEC-1A

MON 2 has 6 other shift functions and we are in the process of writing more software for further functions.

By the time this issue is released, we will have completed this writing and will include documentation with the chip.

The cost of the TEC has risen to $98.00 and it looks like going even higher as the exchange rate for the Aust. dollar drops. But we want to keep the computer below the magical $100 mark for as long as possible.

We have now supplied over 1,000 computers, in 3 different models. Only the earliest model has been fully documented. The upgraded versions vary only slightly and you should have no difficulty constructing them.

The reason for this is the simplicity of the board. Everything is fully identified on the overlay and requires only simple assembly.

Chances are it will operate first go but there is always a small possibility that something will be overlooked and it will not come on.

If you are caught in this situation, here is a run down on how to go about fixing it:

You will need a LOGIC PROBE and a MULTIMETER. A CONTINUITY TESTER (to be presented in next issue) will also be handy.

Firstly the visual checks:

If the displays fail to light-up and no sound is heard from the speaker, the most likely fault will be a broken track or poor solder connection. Turn the computer off and check each track with a multimeter switched to LOW-OHMs.

The regulator should get quite hot and should have 5v on the output lead. It must have at least 8v on the input lead to prevent voltage 'drop-out'.

The Z-80 will get quite warm, as will the output latch near the edge of the board.

The jumper near pin 1 of each latch should be checked. Only one must be inserted for each latch. This means you have two unused holes for each latch.

Check each of the keys for correct positioning. All flats must be DOWN.

The notch on each chip must also be DOWN.

Make sure all the pins of the IC sockets go through the holes in the PC board and are properly soldered. We have seen some pins doubled-up under the socket and not making contact with the tracks.

Check the capacitor near the speed control. It must be 100pf - not 100n. 100pf is indicated by '100' or '101' on a ceramic capacitor whereas 100n is shown as '104' on a mono block or 100nS on a blue body.

Check for non-soldered lands, missing links and incorrectly soldered links. We inspected one project in which the builder had cut the links to the exact length BEFORE soldering and consequently one link did not go through the board completely. It was too short to be soldered but the builder didn't notice. He soldered the land with the result that the link looked as though it was soldered!

Finally check for solder-bridges between adjacent lands with a multimeter set to LOW ohms. Remove the chips to get an accurate reading.

Now for the 'in-depth' diagnosis:

1. Turn the TEC on and check for 5v out of the regulator. Check POWER-ON LED. Check for 5v on each of the chips: 74LS 273 - pin 20. 2716 - Pin 24. 6116 - pin 24. Z-80 - pin 11. 4049 - pin 1 74LS138 - pin 16. 74LS923 - pin 20.

2. Check clock frequency by putting logic probe onto pin 6 of Z-80.

3. Check RESET pin of Z-80 is HIGH.

4. Check NMI line. (pin 17 of the Z-80). It will go LOW when a key is pressed. If not, a switch may be faulty or the keyboard scan oscillator may not be working. Keyboard oscillator is part of the 74C923 and the frequency-setting capacitor and debounce cap are the 100n and 1uf electrolytic.

5. Check pin 19 of the Z-80 with a logic probe. If it is not pulsing, program is not getting through.

6. Logic probe pin 18 of the 2716. Pulses on this pin show the ROM is being accessed.

7. Pulses on pin 18 of the 6116 show RAM is being accessed.

8. No pulses via checks 5, 6 or 7 indicate the full byte in an instruction is not getting through. This may be due to a faulty address or data line.

9. Check Do (pin 9 on the 2716, for continuity to pin 9 of the 6116 and also pin 14 of the Z-80.) Check the other 7 data lines for continuity and also the 11 address lines.

10. With all chips still in circuit, check each pin with the one adjacent to it, for the 2716, 6116 and Z-80. Our continuity checker in issue 14 will be ideal but if you can't wait, a multimeter can be used. Remember protection diodes are contained in most chips and low value resistors may be present on some lines. Low values of resistance may be perfectly acceptable - you are looking for zero ohms or short-circuits between tracks.

11. Check pin 20 of the Z-80 - the IN/OUT REQUEST line. If it is not pulsing, the output of the computer may be putting a load on the data bus.

12. Remove the two output latches and place the negative lead of a continuity tester on one of the pins. Touch every other pin of the output latch with the other lead. Move the first lead and repeat until all pins have been tested. Do the same with the other latch.

This will check for shorts on the data bus as well as between pins of the display.

13 If these fail to locate the fault, ring us at TE. We may be able to help you over the phone. If not, send the TEC in a jiffy padded bag and we will see what the trouble is.

So far we have had about 20 TECs sent in for checking and repair. About 8 of them suffered from voltage surges. This occured when the constructor shorted leads together and/or dropped a screwdriver on the back of the board when the TEC was operating. This can damage the EPROM, RAM and even the Z-80.

Don't let leads from the 'add-ons' dangle over the rest of the computer or let the SELECT leads touch each other when fitting them over the pins on the PC board.

The TEC is really very robust and we haven't damaged a unit yet, even though we have three in constant use and they are let running both day and night.

If you are careful with construction the TEC will work. But as with all pieces of electronic equipment, excess voltage will sound a death knoll.

While on this subject, we repaired two more unusual faults this month.

Both problems were the same and occured like this:

When the constructor was building the TEC, one or more of the components were soldered without being fully pushed onto the board.

Some time later the constructor discovered the fault and proceeded to push the component into place while trying to resolder the joint.

The result was the land broke away form the copper track and created a hairline fracture which was not spotted.

If this occurs on either the address or data bus, the TEC will fail to come on.

If this happens, the first pin to check is each of the Chip Enable pins on the two output latches.

If a probe on these pins show they remain HIGH, they are not being accessed.

Next check the IN/OUT select chip (below the expansion port) and see if it is being activated by the Z-80. No information on pin 4 could indicate that the program is not getting to the Z-80.

This leads you to suspect either one of the data lines or one or more of the address lines. They may be broken, with the result that the Z-80 is not receiving a full byte of program.

Before you jump to this conclusion, check the Chip Enable pin of the EPROM (pin 18) and see that it is LOW. This will mean the 2716 is being accessed and it should be talking to the Z-80.

If the Chip Enable pin is HIGH, go to the ROM/RAM decoder (below the clock chip) and check pin 4 to see that the pin is being accessed.

If one bus line is missing, the Z-80 will get the wrong op-codes and the program will not flow correctly.

Before we continue with programming, here are a few notes on assembling the TEC-1B as some changes have been made since the original notes in issue number 10.

The regulator is placed under the PC and bolted to the board via a 6BA nut and bolt. You can add heat fin if a number of add-ons are to be driven, but under normal circumstances, the regulator and board will dissipate the 1½ to 2 watts of heat.

The electrolytic has been changed to 1000mfd 25v and it lays flat on the board to keep a low profile.

The display drivers are slim-line types and 3 alternatives have been allowed for in the PC pattern. The overlay shows which links are to be added for the type chosen. Only ONE link must be used for each chip.

Finally a Z-80 or Z-80A can be used as the CPU chip. We are operating the TEC at 100kHz to 500kHz and this is well below the maximum speed for either type. A Z-80 will operate up to 2.5MHz and Z-80A up to 4MHz.

If any of the keys become worn, their contacts become erratic and sometimes a double-entry occurs. This can be overcome by increasing the value of the 1mfd on the 74c923 keyboard encoder to 4.7mfd or even 10mfd. This will mask out the contact bounce and produce a single pulse.

A 100n up to 10mfd can be used across the reset and it may be necessary to use the higher value if the Z-80 does not reset properly.

A 10k or 20k cermet can be used as the speed control and it can be either a VTP or HTP type. The advantage of a cermet means you can use your fingers to turn the pot and don't require a small screwdriver.

SHIFT

The latest addition to the TEC software is a SHIFT function.

This enables the number of functions to be increased from 4 to 24.

It means each of the buttons can be programmed to perform a second function when combined with the SHIFT button.

To access this second function the SHIFT button must be pressed first and kept pressed while the desired key is pressed.

HOW DOES IT WORK?

The keyboard encoder uses 5 lines of the data bus and the remaining 3 lines are not used.

The SHIFT button is connected to one of these lines and the monitor program re-written to detect its status when the keyboard is read.

Five functions are currently available. More are in the pipeline and their details will be explained in future articles.

The 5 functions are:

SHIFT +

This is the **INSERT** function. It moves every byte in the program up to the next higher location and inserts **00** into the present address. This operation can be repeated any number of times to produce empty locations.

We have mentioned MON 2 allows programming to start at **0900** and the shift function operates in the area **0900** to **4000.** Addresses above **4000** are not catered for by the software but can be included if required.

Addresses below **0900** may cause a systems crash if you try to insert in this area as it is reserved for scratch pad, pointers and stack etc. Data below **0800** cannot be shifted as it is in ROM.

SHIFT – (shift, MINUS)
This is the **DELETE** key. It performs the opposite of INSERT. The data at the address currently being displayed is removed and all data above this address (and below **4000**) will be shifted DOWN one location. **3FFF** is loaded with **00.**

SHIFT ADdress
This function enables you to jump quickly to a particular location. Suppose you require to address **0A00** on a number of occassions. By pressing **SHIFT ADdress** the micro will jump to **0A00.** For this to happen, you must load a pointer location with the value **0A00,** then every time the SHIFT ADdress buttons are pressed, the display will show **0A00.** The pointer area is two bytes of memory located at **08D2 and 08D3.** By placing the **JUMP ADDRESS** at this location, the operation will be carried out.

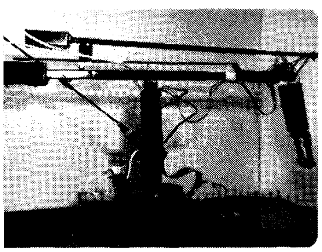
We are loading these two locations directly into BC register pair via a 4-byte instruction **ED 4B D2 08** and for the register pair to be correctly loaded, we must place the lower byte first in memory and then the high byte. This means we must load location **08D2** with **00** and **08D3** with **0A.**

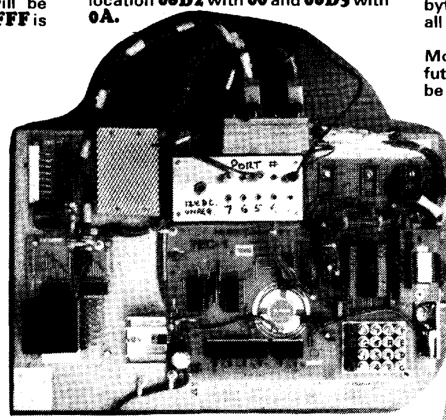
SHIFT 3
This function works exactly like SHIFT ADdress and enables you to have a second address to jump to. This time the pointer area is at **08D4** and **08D5.**

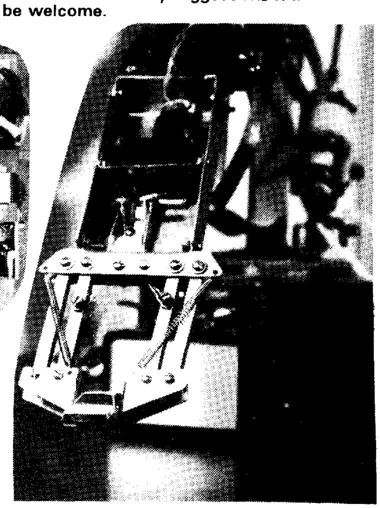
SHIFT 0
This is a search function. If you want to locate a value in a program or table, you could step through until it is located. This could take a long time. But with this function the value can be found very quickly. You can also locate the address of every other time it appears in a program.

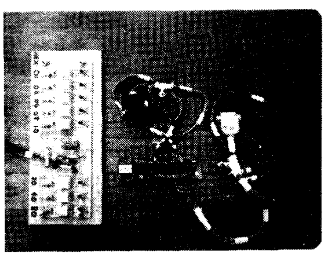
The value of the byte you are looking for is placed at **08E1.** Address the program you are testing and push **SHIFT 0.** The display will illuminate with the address of the byte you are looking for. Pushing SHIFT 0 again will display the second address of the byte. This can be continued to locate all the addresses.

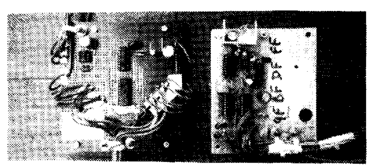
More function will be inclused in future monitors. Any suggestions will be welcome.

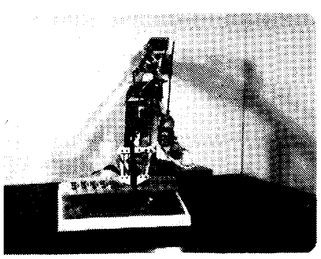


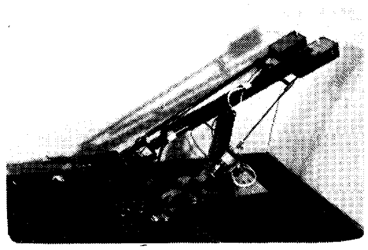


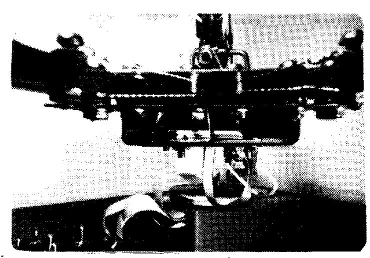












These photos show our science/electronics/computer teacher's add-ons to the TEC and Glen Robinson's Robot Arm. It is made entirely from easy-to-obtain hardware parts, gears, motors and sturdy pieces of steel. A larger photo will do it more justice and this we will show in the next issue.

SIMPLIFYING PROGRAMS

One of the most important features of machine code is the fact that it occupies the least amount of memory.

The skill is to make use of this fact.

If we take the simple program from issue 12 page 21 (1st column), **RUNNING SEGMENT A ACROSS THE SCREEN**", we can shorten the program by using the following set of instructions:

```
LD A,01        800    3E 01
OUT (02),A     802    D3 02
OUT (01),A     804    D3 02
PUSH AF        806    F5
LD DE 20FF     807    11 FF 20
DEC DE         80A    1B
LD A,E         80B    7B
OR D           80C    B2
JR NZ 080A     80D    20 FB
POP AF         80F    F1
RLCA           810    07
JR 0804        811    18 F1
```

This program saves 8 bytes but has the disadvantage that the delay routine cannot be used by any other programs as it is hidden in the listing.

The delay could be placed apart if desired.

Eight bytes may not seem many to save but is a start to efficient programming.

This is where the byte-saving occured:

The instruction RLCA is a one-byte instruction to shift the contents of the accumulator left. (It does not shift through the carry bit but sets it, as explained in data sheet 13.)

The listing contains a number of **JR** instructions (and a displacement byte). These are 2 byte instructions whereas a **CALL** instruction requires 3 bytes.

THE DISPLACEMENT BYTE.

As listings get longer and more complex, the value of the displacement byte requires a method for determining its value.

When the jump is 5, 10 or 15 bytes forward or backward, the displacement value can be obtained by counting the locations: such as 00, 01, 02, 03 or FE, FD, FC, FB, FA etc. But when the jump is 20, 30 or more locations, the value can be obtained via a simple mathematical procedure.

Determining the value of the displacement byte requires 6 steps. By following these you cannot make a mistake.

Step 1. Count, via normal counting, the number of bytes between the displacement byte and the location being jumped to. Include the location you wish to land on. e.g: Take the following example:

11 FF 20
1B
7B
B2
20 dis

The number of bytes between **dis** and **1B** are: **20, B2, 7B, 1B.** These are counted as 1, 2, 3, 4. Thus the answer is 4.

We will select a higher value for our problem to emphasise the need for the procedure.

Suppose the number of locations we wish to jump back is 49.

Step 2: Convert 49 to a HEX value by dividing it by 16:

The answer is 31H

Step 3: Convert 31H to binary:

```
   3       1
0011    0001
```

Step 4: Change each 1 to 0 and each 0 to 1:

Ans: 1100 1110

Step 5: Add 1 to the answer:

Ans: 1100 1111

Step 6: Convert to a HEX value:

C F

This is the value of the displacement byte required to achieve a backward jump of 49 bytes.

The machine code instruction will depend on the **JR** condition and will be one of the following:

28 CF, **20 CF,** or **18 CF**

The steps we have performed are called **TWO S COMPLEMENT.**

Using the knowledge we have gained, we will improve the **BACK and FORTH** program from P 15 of issue 12.

Mainly aiming at byte reduction, we will include a **BIT TESTING** instruction to prevent overshoot of the displays. Bit 0 in the accumulator is tested and if it is a '1', the program will cause a change in direction by rotating the accumulator in the opposite direction.

With these alterations in the program we will save about 12 bytes. Try the program:

```
LD A,01        800    3E 01
OUT (02),A     802    D3 02
OUT (01),A     804    D3 01
CALL DELAY     806    CD 00 0A
RLCA           808    07
BIT 6,A        809    CB 77
JR Z 0804      80B    28 F6
RRCA           80D    0F
OUT (01),A     80E    D3 01
CALL DELAY     810    CD 00 0A
BIT 0,A        813    CB 47
JR Z 080D      815    28 F6
JR 0809        817    18 F0
```

at 0A00:

F5
11 FF 20
1B
7B
B2
20 FB
F1

The program is required to test bit 6 in the accumulator. If it is found to be a '1', the contents of the accumulator is shifted in the opposite direction. Bit 0 is then tested and when found to be '1', the program jumps back and shifts the accumulator in the original direction.

BYTE TABLE. To use this table, the byte following the JR instruction is counted as BYTE ZERO. From this byte you count in either the positive or negative direction using decimal counting.

0	00	48	30	96	60	-1	FF	-49	CF	-97	9F
1	01	49	31	97	61	-2	FE	-50	CE	-9B	9E
2	02	50	32	98	62	-3	FD	-51	CD	-99	9D
3	03	51	33	99	63	-4	FC	-52	CC	-100	9C
4	04	52	34	100	64	-5	FB	-53	CB	-101	9B
5	05	53	35	101	65	-6	FA	-54	CA	-102	9A
6	06	54	36	102	66	-7	F9	-55	C9	-103	99
7	07	55	37	103	67	-8	F8	-56	C8	-104	98
8	08	56	38	104	68	-9	F7	-57	C7	-105	97
9	09	57	39	105	69	-10	F6	-58	C6	-106	96
10	0A	58	3A	106	6A	-11	F5	-59	C5	-107	95
11	08	59	3B	107	6B	-12	F4	-60	C4	-108	94
12	0C	60	3C	108	6C	-13	F3	-61	C3	-109	93
13	0D	61	3D	109	5D	-14	F2	-62	C2	-110	92
14	0E	62	3E	110	6E	-15	F1	-63	C1	-111	91
15	0F	63	3F	111	6F	-16	F0	-64	C0	-112	90
16	10	64	40	112	70	-17	EF	-65	BF	-113	BF
17	11	65	41	113	71	-18	EE	-66	BE	-114	8E
18	12	66	42	114	72	-19	ED	-67	BD	-115	8D
19	13	67	43	115	73	-20	EC	-68	BC	-116	8C
20	14	68	44	116	74	-21	EB	-69	BB	-117	BB
21	15	69	45	117	75	-22	EA	-70	BA	-118	8A
22	16	70	46	118	76	-23	E9	-71	B9	-119	89
23	17	71	47	119	77	-24	E8	-72	B8	-120	B8
24	18	72	48	120	78	-25	E7	-73	B7	-121	87
25	19	73	49	121	79	-26	E6	-74	B6	-122	86
26	1A	74	4A	122	7A	-27	E5	-75	B5	-123	85
27	1B	75	4B	123	7B	-28	E4	-76	B4	-124	84
28	1C	76	4C	124	7C	-29	E3	-77	B3	125	83
29	1D	77	4D	125	7D	-30	E2	-7B	B2	-126	82
30	1E	78	4E	126	6E	-31	E1	-79	B1	-127	81
31	1F	79	4F	127	7F	-32	E0	-80	B0	-128	80
32	20	80	50			-33	DF	-81	AF		
33	21	81	51			-34	DE	-82	AE		
34	22	82	52			-35	DD	-83	AD		
35	23	83	53			-36	DC	-84	AC		
36	24	84	54			-37	DB	-85	AB		
37	25	85	55			-38	DA	-86	AA		
38	26	86	56			-39	D9	-87	A9		
39	27	87	57			-40	D8	-88	A8		
40	28	88	58			41	D7	-89	A7		
41	29	89	59			-42	D6	-90	A6		
42	2A	90	5A			-43	D5	-91	A5		
43	2B	91	5B			-44	D4	-92	A4		
44	2C	92	5C			-45	D3	-93	A3		
45	2D	93	5D			-46	D2	-94	A2		
46	2E	94	5E			-47	D1	-95	A1		
47	2F	95	5F			-48	D0	-96	A0		

INTRODUCTION TO COUNTING

A microprocessor system is ideally suited to counting situations. It can be programmed to count to any particular number then sound an alarm or operate a relay or even notify the near-completion of a run.

It can count UP or DOWN as well as count in sub-multiples.

Take the case of packing a box of TE magazines.

Firstly the operator requires a count of 10. Each 10 issues must be placed in opposite directions in a box to produce a level stack. The operator then needs to know when a count of 140 is reached, which represents a full box.

Finally the packers need to know how many boxes of magazines have been packed so that the delivery docket can be filled out.

This is effectively 3 counters which must be interconnected to achived the required result. Ideally an audible signal should be produced at the end of each count of 140 so that the packer(s) can concentrate (day dream) on the job.

The chance of finding such a design is almost nil, except via individual modules which will have to be connected together to create the system. The cost of doing this would be about $300!!

But with a microprocessor system such as the TEC, all these up-down requirements are possible in the one unit, by simply providing a program!

The art of producing a suitable program is the content of this section.

We will start from the beginning and explain how counting is achieved, how to interface a 'count-button' and progress to producing a 3-digit up-down counter.

A count-down system is often used as it can be pre-programmed with a START VALUE and the counter decrements to zero. It then sounds a bell, activates a relay and resets to the pre-determined start-value.

After studying the 3-digit counter you will be able to create a 4, 5 or 6 digit counter and even incorporate sub-values to facilitate packing etc.

The counter can also be designed to have 2 concurrent tallies, one being permanently displayed while the other is available on call-up via the press of a button.

They would be displayed for a few seconds and fall back into memory.

Absolutely any combination, application or requirement can be catered for, it only requires programming.

To make it easy to understand, we have started with a simple program. But, as explained, this type of program soon runs out of capability. Thus a more complex system of time-sharing of the displays must be used.

But this too has limitations and finally an even more complex (as far as understanding is concerned) use of registers, must be employed.

With this high-level system, the scope is enormous. The system can be increased to 8 digits, two or more separate readouts, and have tally values available on call-up.

This is where we start . . .

Creating your own COUNTING MACHINE is one of the capabilities of our micro. You can produce a display which increments or decrements by a count of one or more on each press of a button. And the button doesn't have to be the '1' button. In our case we have used the '4' button to show that any button can be used.

By changing the values in the 'look-up' table, you can create the up or down condition - something which is virtually impossible with discrete counting-chip construction.

You can even produce letters of the alphabet and increment each time 'Z' or 'F' or 'X' appears. You can do anything from counting by 2's to dividing by 'Z'.

For our first exercise we will produce a counter which counts to 9. This is a very simple program. Only one display will be accessed and thus we can output to it so that it turns on HARD, while the computer is in the HALT mode, waiting for an interrupt from the keyboard.

It is important to note the computer does not produce the numbers 0-9, the program creates them. The table at **0900** contains values which turn on various segments of the display to create the numbers.

0-9 COUNTER

LD A,01	800	3E 01	The accumulator is loaded with **01** and outputted to port **01**. This connects the
OUT (1),A	802	D3 01	cathode of the first display to earth.
LD HL,0900	804	21 00 09	Load HL pair with the address of the number table.
LD A,(HL)	807	7E	Load the first byte of the number table into the accumulator.
OUT (2),A	808	D3 02	Connect segments of the display to the positive rail to get first number.
LD B,0A	80A	06 0A	Register B is our 'counting register'. It counts 10 bytes from **0900** to **0909**.
HALT	80C	76	HALT the program so that first number (0) will appear on the display.
CP 04	80D	FE 04	The program recognises only button '4'.
JR NZ Halt	80F	20 FB	If not button '4', go to HALT. If button '4' pressed, increment HL to look at 0901.
INC HL	811	23	The byte at 0901 is loaded into the accumulator.
LD A,(HL)	812	7E	The value at 0901 (**28**) creates the figure '2' on the display.
OUT (2),A	813	D3 02	Output **28** to port 02.
DJNZ Halt	815	10 F5	Register B is decremented and if it is not zero, the program goes to HALT.
JP Z 0800	817	CA 00 08	When register B is zero, the program jumps to START (**0800**).

at 0900

EB	=	0
28	=	1
CD	=	2
AD	=	3
2E	=	4
A7	=	5
E7	=	6
29	=	7
EF	=	8
AF	=	9

Type the program into the TEC and press RESET, GO. The number '0' will appear on the display.

Press various buttons on the keyboard and notice that only button '4' advances the count.

Step through the table by pressing button 4.

1. Experiment with the program by creating the numbers on another display.
2. Create a down-count by inserting the table at **0900** in the opposite direction. i.e: **AF, EF, 29, E7, A7, 2E, AD, CD, 28, EB.**
3. Create a count-to-six by changing the value of **B** (**080A**) to **06.**
4. Create the letters A-F by adding their appropriate hex values to the table, select the correct value for **B**, change the compare value to enable button 'C' to operate and step through the table you have produced.

TWO DIGITS

When two or more digits are to be displayed, the program must contain a multiplexing or time-sharing arrangement so that each display can show a number from 0 to 9 without interfering with the other. This means a HALT instruction cannot be used as only one display will remain alight!

The program must be constantly looping or 'running' so that both displays are kept on. Each time the program cycles, it is looking for an interrupt from the keyboard and if one comes along, the program operates on the data it receives and compares it with the value 04. Depending on the result, the program will branch to one of two places.

The program below produces a count-to-99 using the '4' button as the input.

The basic structure of the program is quite simple and uses register pair HL to point to the address (at 0900) for the hex value needed to produce the numbers 0 to 9.

Register pair DE points to the hex value (again at 0900) needed to produce the 10's value.

Each of these register pairs are incremented and compared with FF to see if the end of the table has been reached. The increment of the DE register takes place when FF is detected on the 1's count. When the 10's count reaches the end of the table, the whole program is reset.

The computer does not know it is counting to 10. It merely knows it is incrementing through a table. You could put Chinese values on the display and count to 11, simply by changing the value of a few locations.

Here is the 0-99 program and an explanation of each step:

0-99 COUNTER

XOR A	800	AF	Set the accumulator to ZERO.
LD I,A	801	ED 47	Load the interrupt register with ZERO.
LD DE,0900	803	11 00 09	Load DE pair with address **0900.**
LD HL,0900	806	21 00 09	Load HL pair with address **0900**. END OF START-UP.
XOR A	809	AF	Beginning of MAIN PROGRAM. Clear Accumulator.
OUT (1),A	80A	D3 01	Turn OFF 1's display.
LD A,(HL)	80C	7E	Load accumulator with byte pointed to by HL pair.
OUT (2),A	80D	D3 02	Output to port 2.
LD A,01	80F	3E 01	Load accumulator with 1.
OUT (01),A	811	D3 01	Output accumulator to port 1. Display is illuminated.
LD B,10	813	06 10	Register B is a COUNT REGISTER. Load it with 10 to create 16
DJNZ FE	815	10 FE	loops to turn on 1's display.
XOR A	817	AF	Clear Accumulator.
OUT (1),A	818	D3 01	Output 0 to port 1 to turn OFF display.
LD A,(DE)	81A	1A	Load accumualtor with byte at 0900 etc as pointed to by DE pair.
OUT (02),A	81B	D3 02	Output the value thus obtained to port 2.
LD A,02	81D	3E 02	Load the accumulator with 2
OUT (01),A	81F	D3 01	Output to port 1 to turn on 10's display.
LD B,10	821	06 10	Load count register with 10 (decimal 16) and create 16 loops to
DJNZ FE	823	10 FE	turn on 10's display.
LD A,I	825	ED 57	Load the interrupt register into the accumulator.
CP 04	827	FE 04	Compare with 4. i.e. subtract 4 from I. If the result is ZERO,
JP NZ 0809	829	C2 09 08	advance to **082C** If the answer is NOT ZERO, go to **0809.**
XOR A	82C	AF	Clear Accumulator.
LD I,A	82D	ED 47	Load the Interrupt register with ZERO.
INC HL	82F	23	Increment register HL to point to address **0901** etc
LD A,(HL)	830	7E	Load the value at **0901** into the accumulator.
CP FF	831	FE FF	Compare the value obtained (eg **28**) with **FF.** If equal, advance to
JP NZ 0809	833	c2 09 08	**0836,** if NOT equal, go to **0809.**
INC DE	836	13	Increment DE.
LD A,(DE)	837	1A	Load the value pointed to by register DE into the accumulator.
CP FF	838	FE FF	Compare with **FF** to see if end of table has been reached.
JP NZ 0806	83A	C2 06 08	If **FF** is reached, result will be zero. Advance to **083D. If not, go to 0806.**
JP 0800	83D	C3 00 08	**JUMP TO START**

at 0900:

EB	= 0
28	= 1
CD	= 2
AD	= 3
2E	= 4
A7	= 5
E7	= 6
29	= 7
EF	= 8
AF	= 9
FF	

The **CONDITIONAL JUMP** instruction requires explanation.

In the 00-99 counter program above, there are three places where the Z-80 will jump to another part of the program when a certain condition is met. The condition is **NZ** (NON ZERO). Let us explain how to interpret this:

From the program above:

LD A,I
CP 04
JP NZ 0809

These 3 lines state: The I register is loaded into the accumulator. The accumulator is compared with 04. Jump to 0809 is the result is NON ZERO.

How does the COMPARE statement work?

The **CP** operation is carried out like a subtract operation and the zero flag (Z flag) will be SET if the result is ZERO and RESET if the result is NON ZERO. This means it will be '1' if the answer is zero and '0' if the answer is not zero.

This is quite confusing because you have to deal with the negative of a negative. To simplify things we can use the word **MET** for ZERO. Thus we get:

JP NZ 0809 → **NOT 04** / **I = 04**

Jump to **0809** if I is not 04 or go to the next line of the program if I = 04.

Now we come to the THREE DIGIT COUNTER. It has an UP/DOWN facility as well as CLEAR. Push +for increment, − for decrement and push ADdress to zero the display. The counter can also be preset by loading **0B03** and **0B04** with values as shown in the listing on the right:

```
PUSH AF      A00  F5
CALL 0A0D    A01  CD 0D 0A
POP AF       A04  F1
RRA          A05  1F
RRA          A06  1F
RRA          A07  1F
RRA          A08  1F
CALL 0A0D    A09  CD 0D 0A
RET          A0C  C9
```

```
AND 0F       A0D  E6 0F
LD HL        A0F  21 00 0C
ADD A,C      A12  85
LD L,A       A13  6F
LD A(HL)     A14  7E
LD (BC)A     A15  02
INC BC       A16  03
RET          A17  C9
```

THREE DIGIT COUNTER

```
START  LD BC 0B00     800   01 00 0B
       LD DE 0B03     803   11 03 0B
       LD A(DE)       806   1A
       CALL 0A00      807   CD 00 0A
       INC DE         80A   13
       LD A(DE)       80B   1A
       CALL 0A0D      80C   CD 0D 0A
       LD HL 0B02     80F   21 02 0B
       CALL SCAN      812   CD 00 09
       LD A,I         815   ED 57
       LD HL 0B03     817   21 03 0B
INC    CP 10          81A   FE 10
       JRNZ DEC       81C   20 0D
       LD A(HL)       81E   7E
       INC A          81F   3C
       DAA            820   27
       LD (HL)A       821   77
       JRNC START     822   30 20
       INC HL         824   23
       LD A(HL)       825   7E
       INC A          826   3C
       DAA            827   27
       LD (HL)A       828   77
       JR CLEAR       829   18 19
DEC    CP 11          82B   FE 11
       JRNZ RESET     82D   20 0D
       LD A(HL)       82F   7E
       DEC A          830   3D
       DAA            831   27
       LD (HL)A       832   77
       JRNC CLEAR     833   30 0F
       INC HL         835   23
       LD A(HL)       836   7E
       DEC A          837   3D
       DAA            838   27
       LD (HL)A       839   77
       JR CLEAR       83A   18 08
RESET  CP 13          83C   FE 13
       JRNZ CLEAR     83E   20 04
       XOR A          840   AF
       LD (HL)A       841   77
       INC HL         842   23
       LD (HL)A       843   77
CLEAR  LD A,FF        844   3E FF
       LD I,A         846   ED 47
       JR START       848   18 B6
```

SCAN

```
LD B,04      900  06 04
LD A(HL)     902  7E
OUT (02)A    903  D3 02
LD A,B       905  78
OUT (01)A    906  D3 01
LD B 50      908  06 50
DJNZ         90A  10 FE
DEC HL       90C  2B
LD B,A       90D  47
XOR A        90E  AF
OUT (01)A    90F  D3 01
RRC B        911  CB 08
JRNC         913  30 ED
RET          915  C9
```

at 0C00:

```
EB
28
CD
AD
2E
A7
E7
29
EF
AF
```

To make this program easy to understand, we have listed ONE COMPLETE CYCLE. Exactly as it is run by the computer. CALL ROUTINES have been included each time they are called and this makes the listing fairly long.

When the program is run for the first time, the display will show the values contained at **0B03** and **0B04.** For the purpose of showing how the program works, we will place **21** at **0B03** and **43** at **0B04.** This will cause the display to show 123 (the value 4 will not appear in this 3 digit counter).

Follow through each of the steps and you will see how the program picks up data from the 'BUFFER ZONE' and converts it values which can be identified as numbers on the display. This program is being executed at more than 100 times per second!

```
START  LD BC 0B00    Location 0B00 stores the value of the units display
       LD DE 0B03    D is loaded with 0B and E is loaded with 03.
       LD A(DE)      Load two nibbles (21 in our example) into the accumulator.
       PUSH AF       Save the accumulator
       AND 0F        This instruction zero's the high nibble leaving 01
       LD HL 0C00    H is loaded with 0C and L with 00
       ADD A,L       Add 00 to the accumulator to get 01 (01 is from above)
       LD L,A        Load the accumulator (it has 01 in it) into the L register.
       LD A(HL)      Load the value at 0C01 (28) into the accum. (HL is now 0C01)
       LD (BC)A      Load the value from the accumulator (28) into the BC register pair.
       INC BC        Increment the BC register (it will become 0B01).
       POP AF        Fetch the accumulator (value 21) from the stack.
       RRA           Shift the value 21 four places to the right
       RRA           so that the high bits will be transposed
       RRA           with the low bits. The result will be 12.
       RRA
       AND 0F        Remove the 4 HIGH bits to get 02.
       LD HL 0C00    H will be loaded with 0C and L with 00
       ADD A,L       Add 00 to the accumulator to get 02.
       LD L,A        Load 02 into the L register.
       LD A(HL)      Load the value at 0C02 (CD) into the accumulator.
       LD (BC)A      Load the accumulaotr (it has 02 in it) into the address pointed to by BC.
       INC BC        Increment the BC register (to 0B02)
       INC DE        DE is incremented to 0B04.
       LD A(DE)      The value at 0B04 (43) is loaded into the accumulator.
       AND 0F        The HIGH nibble is cleared to get 03.
       LD HL 0C00    H is loaded with 0C and L with 00.
       ADD A,L       00 Is loaded into the accumulator to get 03.
       LD L,A        03 is loaded into L.
       LD A(HL)      The vlaue at 0C03 'AD) is loaded into the accumulator.
       LD (BC)A      Load AD into location 0B02.
       INC BC        The BC register pair is incremented to 0B03
       LD HL 0B02    Load H with 0B and L with 02.
       LD B,04       Load B with 04.
       LD A(HL)      Load the accumulator with the value at 0B02. (AD).
       OUT (02),A    Output AD to port 02.
       LD A,B        Load the accumulator with 04.
       OUT (01),A    Output 04 to port 01. This will turn on a,b,c,d,g to get '3'.
       LD B,50       B is loaded with 50hex (five-oh or 80 in decimal)
       DJNZ          Perform a jump command for 80 loops.
       DEC HL        HL now points to 0B01.
       LD B,A        The accumulator (it contains 04) is loaded into B.
       XOR A         Clear the accumulator.
       OUT (01),A    Turn OFF the display
       RRC B         Shift register B right to get 02 (half its previous value).
       LD A(HL)      Load the value at 0B01 (CD) into the accumulator.
       OUT (02),A    Output the value CD to port 2.
       LD A,B        Load B (02) into the accumulator.
       OUT (01),A    Output 02 to port 1. This turns on the second display and a,b,d,e,g: '2'.
       LD B,50       Load B with 50 (in hex)
       DJNZ          Perform 50 loops. This is 80 loops.
       DEC HL        HL now points to 0B00.
       LD B,A        Load 02 into B.
       XOR A         Zero the accumulator.
       OUT (01),A    Turn OFF the display.
       RRC B         Rotate register B to the right to get 01.
       LD A(HL)      Load the value at 0B00 (28) into the accumulator.
       OUT (02),A    Output 28 to port 2.
       LD A,B        Load 01 into the accumulator.
       OUT (01),A    Output 01 to port 1.
       LD B,50       Load B with 50.
       DJNZ          This instruction creates 80 loops of delay-time.
       DEC HL        HL is decremented but the 4th loaction is not used as you will see.
       LD B,A        Load 01 into B.
       XOR A         Zero the accumulator.
       OUT (01),A    Turn off the display.
       RRC B         Register B is shifted and the carry bit is SET.
       LD A,I        The accumulator is loaded with a value from the keyboard.
       LD HL 0B03    H is loaded with 0B and L with 03.
INC    CP 10         The value 10 is compared with the accumulator.
       DJNZ          If the two are the SAME, the program increments. If not it jumps to DEC.
       LD A(HL)      Load A with 21.
       INC A         Increase the value 21 to 22.
       DAA           Decimal adjust the accumulator if needed (not in this case).
       LD (HL),A     Load 22 into the location 0B03.
       JRNC start    Jump to start is no carry from DAA operation. If a carry is produced, i.e.
       INC HL        when 99 advances to 100. increment HL to 0B11.
       LD A(HL)      Load the value at 0B11 into the accumulator.
       INC A         Increment A.
       DAA           Decimal adjust the accumulator if necessary.
       LD (HL)A      Load the accumulator into 0B04.
       JR CLEAR      Jump to CLEAR
CLEAR  LD A,FF       Load FF into the accumulator.
       LD I,A        Load the accumulator into the interrupt vector register.
       JR START      Jump to START.
```

TEC-1A & 1B

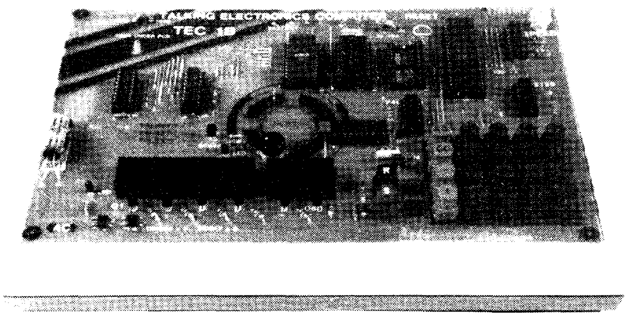
Kit of parts: $90.60
PC Board: $24.30
Complete: $114.90

TALKING ELECTRONICS COMPUTER

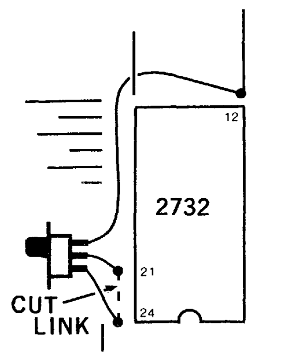
TEC 1A's can be converted to TEC 1B's by ading a push button, a 47k resistor and a diode. When you update to MON 2, the SHIFT function allows INSERT and DELETE and a number of other commands.

PART V

Features in this article:
★ **Crystal Oscillator**
★ **Input/Output Module**



TEC 1B with SHIFT KEY FITTED.



This is the fifth article on the TEC and quite frankly we have only just scratched the surface up to now.

The more ideas you try, the more you realise the potential of programming.

We have received a number of programmes for the 7-segment displays as well as the 8x8. These have been included in this article and also a few more hints on programming in general.

But before we get onto the programmes, there are a number of loose ends we have to tidy up, to bring the documentation up to date.

So far there have been 4 different models of the TEC and although the changes have been slight, they have not been put down on paper.

As far as the software is concerned, all models are compatible as the only modifications have been in the hardware.

The output latches have been changed from 8212's to 74LS273's, the 2200uF filter electrolytic changed to 1000uF and the 7805 mounted under the board so that its leads cannot be bent or broken.

The rest of the design remains substantially the same with the only addition being a shift button near the keyboard.

This button allows the keys to have a second function and we have already described these in issue 13.

Kits are now supplied with both the 1B ROM and also MON 2 ROM. It is possible to fit both programs into a single 2732 and to select either one program or the other requires a slide switch to take pin 21 HIGH or LOW. With this you can get the best of both monitors.

The computer can be switched between one MONitor and the other by pressing the reset button and while it is pressed, the slide switch is changed. When the reset button is released, the other MON will come into operation.

The following is a reprint of an information sheet supplied with the latest kits:

THE 2732 MONITOR

Both MON 1B and MON 2 are in the same chip and is called MON 1B/2. The MON 1B program has been placed in the upper half of memory so that when it is placed in the TEC, the MON 1B section will run and the computer will display 0800. You can now access all the games, tunes and running letter routines as covered in issues 10, 11, 12 and 13.

The MON 2 routine is more advanced and does not contain any of the games. Instead it has a SHIFT routine that enables you to insert bytes into a program by shifting all the higher bytes, and the byte at the present address, up one location. And a delete function, as well as a number of other routines that have been covered in issue 13.

When you want to accesss the MON 2 program, a switch must be fitted to the board so that pin 21 can be taken to ground. This will enable the lower half of the 2732 to be brought into the system and thus run the MON 2 listing.

The diagram above shows how to fit the mini slide switch to the two halves of the link that has been cut as shown.

You can switch from one monitor to the other at any time by pressing reset and altering the switch.

If you are writing a program using the MON 1B, it is best to start at 0900, so that when (if) you want to use the INSERT or DELETE functions, you can change to MON 2, use the function and then change back to MON 1B.

Gradually you will realise it is best to use MON 2 for most of your programs.

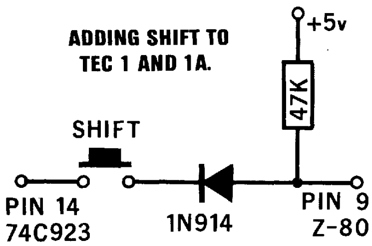
There are two major differences between MON 1B and MON 2. MON 1B uses a simple routine that places the value of a key directly into the accumulator, without firstly saving the value of the accumulator. Thus its original value is destroyed. MON 2 loads the key value into location **08E0** and thus your program must include an instruction that looks at this location for the value of the key.

Unless you load directly into the A register.

Simple programs designed for MON 1B will not run on MON 2 if they include a key press; unless they are altered accordingly.

The second difference is the start address for programming. MON 1B starts at 0800, while MON 2 starts at 0900. Programs written at 0800 cannot be successfully modified via the insert and delete functions as they will run into part of the scratchpad area for the MON 2 system.

The following diagram shows how to add the diode and resistor for the shift function. The diagram in issue 13 was not clear and this is an improvement:



TEC 1A/1B CONSTRUCTION HINTS:

The output latches for the latest TEC's are 74LS273's and the dotted link below each chip is fitted.

The 7805 regulator bolts directly under the PC board and a little thermal compound can be applied to assist heat transfer.

The small link from pin 4 of the 74LS138 IN/OUT decoder must be added. It can be cut later if expansion is required.

About 58 empty holes will be on the board after construction. Some provide for expansion while others are unused.

After the keys have been added and everything is operating satisfactorily, the letters and numbers can be applied to the tops.

Firstly clean the buttons with methylated spirits and apply the rub-down letters. Cover them with clear nail varnish to protect them. If you want to add another layer, wait for the first to dry, otherwise the letters will smudge!

NOTES ON THE 8x8 DISPLAY

The 8x8 has been modified to include sinking and sourcing transistors as described on P 27 of issue 12 and all kits now include 16 transistors and the necessary current limiting resistors.

This results in the LEDs being driven harder and increases the brightness of the display noticeably.

This is important when multiplexing as each LED will be turned on for only about one-eighth of the time and if sufficient current is supplied during this instant, the LED will appear to be on for the total period of time with an acceptable brightness.

We had an interesting fault in an 8x8 last week. It is interesting because the knowledge we gained applies to other projects where LEDs are driven in parallel.

A constructor built the 8x8 and was not happy with the output of about 3 of the LEDs.

He went to his local electronics shop and bought a few replacements.

After fitting them, he was quite surprised that they did not work at all! So he rang us. At this particular point in time we were not familiar with the fault and did not know how to advise him. So we suggested he call around with the project.

Some time later that day he arrived and we noticed the first difference was the colour of the LEDs he had used. They were less opaque than the rest and the crystal inside the LED could be readily seen. This did not disturb us as the light output of the LEDs was our prime concern.

When we tested it, sure enough; the 3 LEDs did not light up.

On measuring across the new LEDs with a multimeter set to low ohms, the voltage drop across the crystal was slightly higher than the rest. (When we are taking a measurement like this, the swing of the needle is being taken as a voltage drop. We are using the 3v suppy in the multimeter to provide the LED with voltage and the needle tells us the characteristic voltage drop across the crystal.)

We then got three LEDs from our stock and measured the characteristic voltage drop. It was exactly the same as the majority in the display and when we fitted them, the whole screen lit up perfectly.

The reason why the LEDs failed to illuminate was due to the higher voltage needed to turn them on. Even if this is 100mV or so, the result will be the LED will not turn on at all. (See the experiment in Stage-1, P 9)

It is important that LEDs are matched according to this characteristic voltage, for situations where they are placed in parallel. The 8x8 is one example as the LEDs are effectively in parallel when the whole screen is being illuminated in a non-multiplexed situation.

DISPLAYING LETTERS AND NUMBERS

The 7-segment display is quite a unique unit. It will display all the numbers from 0 to 9 as well as many of the letters of the alphabet.

There are only about seven letters that cannot be readily displayed and for these we will have to make a compromise.

The letter M is displayed as a small 'n', with a bar over the top. This corresponds to a feature in mathematics where a dot is placed over the first and last digits in a number to indicate the number repeats. (This is called a recurring number or recurring fraction).

The letter W is displayed as a small 'u' with a bar over the top, for the same reason. The letter 'U' is displayed as a capital letter while V is a small 'u'.

The letter 'X' is displayed as part of a cross and Z is shown as two angles in opposite corners of the display, and looks quite readable.

The only letters which require interpretation are 'K' and 'Q'.

Ten other characters have also been included such as a question mark and 'equals' as well as a reverse bracket to assist in displaying mathematical problems.

A	6F
B	E6
C	C3
D	EC
E	C7
F	47
G	E3
H	6E
I	28
J	E8
K	67
L	C2
M	65
N	6B
O	EB
P	4F
Q	3F
R	44
S	A7
T	46
U	EA
V	E0
W	E1
X	26
Y	AE
Z	C9

?	4D
=	84
–	04
!	38
,	10
"	0A
;	30
—	20
≡	85
)	0F

1	28
2	CD
3	AD
4	2E
5	A7
6	E7
7	29
8	EF
9	AF
0	EB

TESTING A BLANK 2716 FOR FF's

After erasing an EPROM, such as a 2716, it is wise to make sure it is entirely blank before reprogramming it. The program that follows does just that. It does not inform you of the location or locations that do not contain FF, but rather the screen goes blank and stays blank if a location has not been fully erased.

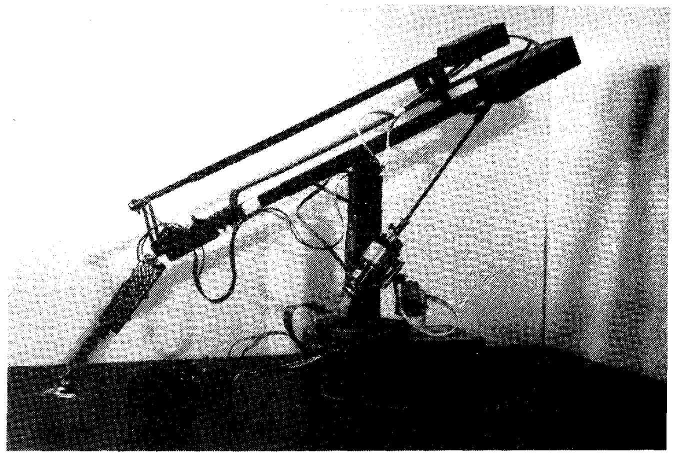
If all locations contain FF, the TEC resets via the MONitor program to the start-up address (either 0800 or 0900). This program can be placed anywhere in RAM and will work with either MON 1 or MON 2.

- by James Doran. 3218

```
11 00 08
21 00 10
7E
FE FF
20 07
23
1B
7A
B3
20 F5
C7
76
```

As promised, a larger photo of the robot arm. If you have built anything like this, why not take a photo and send it in.

Your ideas, combined with others, will help us to present an article.



MON 2 HEX LISTING:

For those with the TEC 1B and an EPROM BURNER, here is the hex listing for the MON 2.

With this you can make your own MON 2, and save the cost of conversion.

Insert the data **0800** on the TEC, and continue through to **0D64.**

Go through the program at least once, checking each of the values to make sure a mistake has not been made. A single mistake can mean the difference between perfection and failure.

MON 2 HEX LISTING FOR TEC 1B:

0000	C3	00	02	FF
0004	FF	FF	FF	FF
0008	2A	C0	08	E9
000C	FF	FF	FF	FF
0010	2A	C2	08	E9
0014	FF	FF	FF	FF
0018	2A	C4	08	E9
001C	FF	FF	FF	FF
0020	2A	C6	08	E9
0024	FF	FF	FF	FF
0028	2A	C8	08	E9
002C	FF	FF	FF	FF
0030	2A	CA	08	E9
0034	FF	FF	FF	FF
0038	2A	CC	08	E9
003C	FF	FF	FF	FF
0040	FF	FF	FF	FF
0044	FF	FF	FF	FF
0048	FF	FF	FF	FF
004C	FF	FF	FF	FF
0050	FF	FF	FF	FF
0054	FF	FF	FF	FF
0058	FF	FF	FF	FF
005C	FF	FF	FF	FF
0060	FF	FF	FF	FF
0064	FF	FF	F5	DB
0068	00	32	E0	08
006C	F1	ED	45	FF
0070	FF	FF	FF	FF
0074	FF	FF	FF	FF
0078	FF	FF	FF	FF
007C	FF	FF	FF	FF
0080	EB	28	CD	AD
0084	2E	A7	E7	29
0088	EF	2F	6F	E6
008C	C3	EC	C7	47
0090	E3	66	28	E8
0094	4E	C2	2D	6B
0098	EB	4F	2F	4B
009C	A7	46	EA	E0
00A0	AC	A4	AE	C9
00A4	10	08	18	04
00A8	2C	00	FF	FF
00AC	FF	FF	FF	FF
00B0	00	09	00	00
00B4	FF	FF	FF	FF
00B8	FF	FF	FF	FF
00BC	FF	FF	FF	FF
00C0	1B	18	1E	1D
00C4	12	17	0E	29
00C8	0B	22	29	17
00CC	12	0C	24	29
00D0	29	29	29	29
00D4	FE	1C	1D	18
00D8	17	0E	FF	FF
00DC	FF	FF	FF	FF
00E0	CD	89	02	03
00E4	18	04	CD	89
00E8	02	0B	CD	90
00EC	04	CD	70	02
00F0	21	DF	08	CB
00F4	C6	CB	8E	C3
00F8	78	03	FF	FF
00FC	FF	FF	FF	FF
0100	FD	10	10	FD
0104	11	EF	12	E1
0108	13	D5	14	C9
010C	15	BE	16	B3
0110	18	A9	19	9F
0114	1A	96	1C	8E
0118	1E	86	20	7F
011C	22	77	24	71
0120	26	6A	28	64
0124	2A	5F	2D	59
0128	2F	54	32	50
012C	35	4B	38	47
0130	3C	43	3F	3F
0134	43	3C	47	38
0138	4B	35	50	32
013C	54	2F	59	2D
0140	5F	2A	64	28
0144	6A	26	71	24
0148	77	22	7F	20
014C	86	1E	8E	1C
0150	96	1A	94	19
0154	A9	18	B3	16
0158	BE	15	C9	14
015C	D5	13	E1	12
0160	EF	11	FD	10
0164	FF	FF	FF	FF
0168	FF	FF	FF	FF
016C	FF	FF	FF	FF
0170	C5	D5	E5	F5
0174	A7	20	03	5F
0178	18	02	1E	80
017C	21	00	01	87
0180	85	6F	4E	23
0184	46	7B	D3	01
0188	10	FE	46	AF
018C	D3	01	10	FE
0190	0D	20	F1	F1
0194	E1	D1	C1	C9
0198	FF	FF	FF	FF
019C	FF	FF	FF	FF
01A0	F5	E5	2A	D6
01A4	08	7E	FE	FF
01A8	20	03	E1	F1
01AC	C9	FE	FE	28
01B0	F1	23	CD	70
01B4	01	18	EE	FF
01B8	FF	FF	FF	FF
01BC	FF	FF	FF	FF
01C0	21	DF	08	CB
01C4	46	20	07	CB
01C8	C6	CB	8E	C3
01CC	78	03	CB	86
01D0	CB	CE	C3	78
01D4	03	FF	FF	FF
01D8	C5	06	80	CD
01DC	A0	02	10	FB
01E0	C1	C9	FF	FF
01E4	ED	4B	D2	08
01E8	CD	90	04	CD
01EC	70	02	C3	78
01F0	03	FF	ED	4B
01F4	D4	08	CD	90
01F8	04	CD	70	02
01FC	C3	78	03	FF
0200	ED	73	E8	08
0204	31	00	09	F5
0208	C5	D5	E5	DD
020C	E5	FD	E5	08
0210	D9	F5	C5	D5
0214	E5	ED	57	F5
0218	AF	32	CC	08
021C	32	CD	08	3E
0220	FF	32	E0	08
0224	C3	40	02	FF
0228	FF	FF	FF	FF
022C	FF	FF	FF	FF
0230	FF	FF	FF	FF
0234	FF	FF	FF	FF
0238	FF	FF	FF	FF
023C	FF	FF	FF	FF
0240	31	C0	08	AF
0244	D3	01	D3	02
0248	21	B0	00	11
024C	D8	08	01	05
0250	00	ED	B0	CD
0254	70	02	3E	08
0258	CD	70	01	3E
025C	0F	CD	70	01
0260	3E	01	32	DF
0264	08	CD	A0	02
0268	CD	60	03	18
026C	F8	FF	FF	FF
0270	F5	E5	C5	CD
0274	89	02	E6	F0
0278	0F	0F	0F	0F
027C	32	DC	08	0A
0280	E6	0F	32	DD
0284	08	C1	E1	F1
0288	C9	21	D8	08
028C	7E	07	07	07
0290	07	23	86	47
0294	23	7E	07	07
0298	07	07	23	86
029C	4F	0A	C9	FF
02A0	F5	E5	D5	C5
02A4	11	D8	08	AF
02A8	D3	01	CD	50
02AC	03	CB	4E	28
02B0	02	C6	E7	D3
02B4	02	3E	20	D3
02B8	01	06	20	10
02BC	FE	AF	D3	01
02C0	CD	50	03	CB
02C4	4E	28	02	CB
02C8	E7	D3	02	3E
02CC	10	D3	01	06
02D0	20	10	FE	AF
02D4	D3	01	CD	50
02D8	03	CB	4E	28
02DC	02	CB	E7	D3
02E0	02	3E	08	D3
02E4	01	06	20	10
02E8	FE	AF	D3	01
02EC	CD	50	03	CB
02F0	4E	28	02	CB
02F4	E7	D3	02	3E
02F8	04	D3	01	06
02FC	20	10	FE	AF
0300	D3	01	00	C3
0304	18	03	FF	FF
0308	FF	FF	FF	FF
030C	CD	89	02	C5
0310	E1	31	C0	08
0314	E9	FF	FF	FF
0318	CD	50	03	CB
031C	46	28	02	CB
0320	E7	D3	02	3E
0324	02	D3	01	06
0328	20	10	FE	AF
032C	D3	01	CD	50
0330	03	CB	46	28
0334	02	CB	E7	D3
0338	02	3E	01	D3
033C	01	06	20	10
0340	FE	AF	D3	01
0344	C1	D1	E1	F1
0348	C9	FF	FF	FF
034C	FF	FF	FF	FF
0450	21	80	00	1A
0354	85	6F	7E	13
0358	21	DF	08	C9
035C	FF	FF	FF	FF
0360	F5	E5	21	E0
0364	08	3E	FF	BE
0368	28	0E	7E	E6
036C	1F	CB	6E	20
0370	02	C6	14	C3
0374	A8	03	FF	FF
0378	E1	F1	C9	FF
037C	FF	E1	F1	C9
0380	FF	FF	FF	FF
0384	CD	89	02	C5
0388	DD	E1	DD	23
038C	DD	E5	E1	7C
0390	FE	40	28	08
0394	DD	7E	00	DD
0398	77	FF	18	EE
039C	3E	00	32	FF
03A0	3F	CD	70	02
03A4	C3	78	03	FF
03A8	C6	01	CD	70
03AC	01	C3	21	04
03B0	CD	89	02	0B
03B4	DD	21	FE	3F
03B8	DD	7E	00	DD
03BC	77	01	DD	2B
03C0	DD	E5	E1	79
03C4	BD	20	F1	78
03C8	BC	20	ED	DD
03CC	36	01	00	CD
03D0	70	02	C3	78
03D4	03	FF	FF	FF
03D8	E5	F5	DD	E5
03DC	C5	AF	32	DF
03E0	08	06	06	21
03E4	D8	08	3E	29
03E8	77	23	10	FC
03EC	2A	D0	08	7E
03F0	FE	FF	20	06
03F4	C1	DD	E1	F1
03F8	E1	C9	FE	FE
03FC	28	EE	DD	21
0400	D8	08	06	05
0404	DD	7E	01	DD
0408	77	00	DD	23
040C	10	F6	7E	32
0410	DD	08	23	06
0414	40	CD	A0	02
0418	10	FB	18	D3
041C	FF	FF	FF	FF
0420	FF	D6	01	36
0424	FF	CB	67	C2
0428	C0	04	CB	6F
042C	C2	C0	04	21
0430	DF	08	CB	46
0434	CA	55	04	57
0438	CD	89	02	21
043C	DF	08	CB	5E
0440	20	03	AF	CB
0444	DE	07	07	07
0448	07	E6	F0	82
044C	02	CD	70	02
0450	C3	7D	03	FF
0454	FF	57	21	DF
0458	08	CB	9E	CB
045C	66	20	08	01
0460	00	00	CD	90
0464	04	CB	E6	CD
0468	89	02	78	07
046C	07	07	07	E6
0470	F0	5F	79	07
0474	07	07	07	E6
0478	0F	83	47	79
047C	07	07	07	07
0480	E6	F0	82	4F
0484	CD	90	04	CD
0488	70	02	C3	7D
048C	03	FF	FF	FF
0490	F5	E5	21	D8
0494	08	78	E6	F0
0498	07	07	07	07
049C	77	23	78	E6
04A0	0F	77	23	79
04A4	E6	F0	07	07
04A8	07	07	77	23
04AC	79	E6	0F	77
04B0	E1	F1	C9	FF
04B4	FF	FF	FF	FF
04B8	FF	FF	FF	FF
04BC	FF	FF	FF	FF
04C0	21	DF	08	CB
04C4	9E	CB	A6	FE
04C8	10	CA	E0	00
04CC	FE	11	CA	E6
04D0	00	FE	12	CA
04D4	0C	03	FE	13
04D8	CA	C0	01	FE
04DC	14	CA	50	05
04E0	FE	15	CA	FF
04E4	FF	FE	16	CA
04E8	FF	FF	FE	17
04EC	CA	F2	01	FE
04F0	18	CA	70	05
04F4	FE	19	CA	FF
04F8	FF	FE	1A	CA
04FC	FF	FF	FE	1B
0500	CA	FF	FF	FE
0504	1C	CA	60	06
0508	FE	1D	CA	FF
050C	FF	FE	1E	CA
0510	FF	FF	FE	1F
0514	CA	FF	FF	FE
0518	20	CA	FF	FF
051C	FE	21	CA	FF
0520	FF	FE	22	CA
0524	FF	FF	FE	23
0528	CA	FF	FF	FE
052C	24	CA	B0	03
0530	FE	25	CA	84
0534	03	FE	26	CA
0538	FF	FF	FE	27
053C	CA	E4	01	C3
0540	78	03	FF	FF
0544	FF	FF	FF	FF
0548	FF	FF	FF	FF
054C	FF	FF	FF	FF
0550	CD	89	02	60
0554	69	3A	E1	08
0558	23	BE	20	FC
055C	44	4D	CD	90
0560	04	C3	53	02
0564	FF	FF	FF	FF

HOW THE CIRCUIT WORKS
(and a general discussion.)

The circuit diagram is TALKING ELECTRONICS COMPUTER 1B (TEC 1B). It is a 9-chip, single-board computer capable of executing Machine Code commands and displaying the result on either the inbuilt display (a set of 7-segment displays) or on other displays via the expansion socket.

The expansion socket is configured identical to the RAM socket and is accessed via line Y2 of the ROM/RAM decoder 74LS138, at the top right-hand corner of the diagram.

The computer starts-up via a MONitor program contained in the 2732 and two monitor programs are in this chip.

The MON 1 select switch takes address line A11 LOW for the low half and HIGH for the upper half.

The other major change between TEC 1 and TEC 1B is the output latches. They were originally 8212's but now 74LS273's have been used. These are a modern chip and are more readily available.

STARTING UP

When the power is applied to the computer, the reset line on the Z-80 is taken low for an instant via the 100n capacitor and this resets the internal workings of the Z-80.

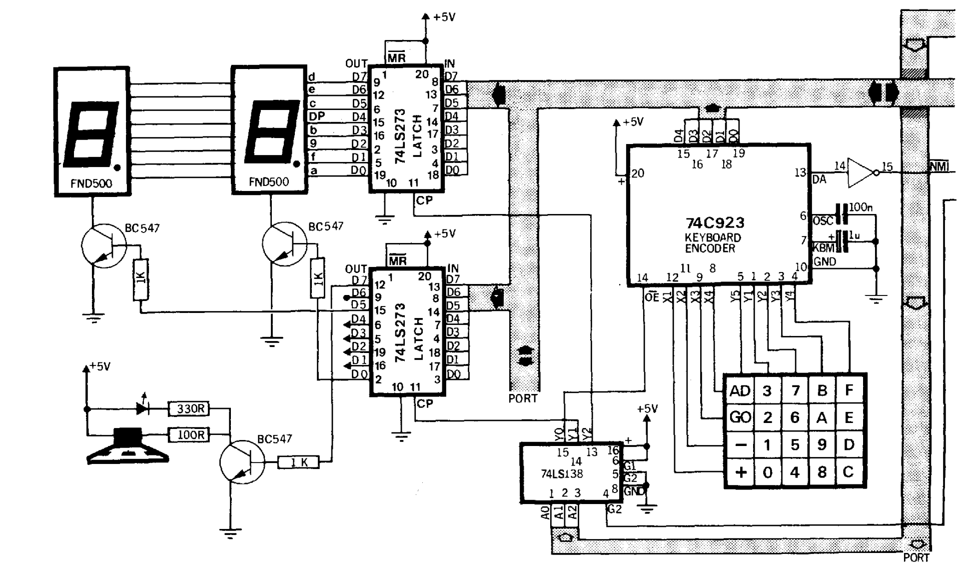
Its first operation is to look for the first byte of data at address zero, in the monitor. Depending on this being a one-

contains 11 lines while the data bus contains 8 lines. The data bus is always 8 bits wide for a Z-80 processor and this gives it the name '8-bit system'.

The address bus is a **ONE-WAY** bus in which the Z-80 activates the lines and turns them on and off using binary notation to generate an address value.

When all lines are LOW, address zero is represented. When line A0 is HIGH, address 1 is represented. The Z-80 has 16 address lines and address 1 is:
0000 0000 0000 0001. When line A1 is HIGH, address 2 is:0000 0000 0000 0010

The address lines connect to a number of chips but only one will respond due to a 'turn-on' line called a command line being required to be activated.



TEC 1B COMPUTER CIRCUIT

When the ROM select switch is HIGH, MON-1 program is accessed and the computer displays **0800**. When the switch is LOW, the computer displays **0900** and the MON 2 program operates.

This has been done so that the TEC 1B is compatible with the original TEC 1 and it can be upgraded by adding a monitor switch and a programmed 2732 EPROM.

The original TEC 1 had a 2716 EPROM but these chips are no longer manufactured and thus a 2732 is now used. When a 2732 is placed in a 2716 socket the upper half of the chip is accessed and thus MON 1 program has been placed in the upper half.

byte, two-byte or three-byte instruction, the Z-80 will execute it or request one or two more bytes.

The flow of information from the Z-80 to the other chips is via two buses. They are the **ADDRESS BUS** and **DATA BUS**. In addition, there is a set of control lines (sometimes referred to as the control bus) that activate (generally) one chip at a time.

All signals within the computer are at a level equal to rail voltage (called HIGH) or ground (called LOW). For this reason they are called digital circuits.

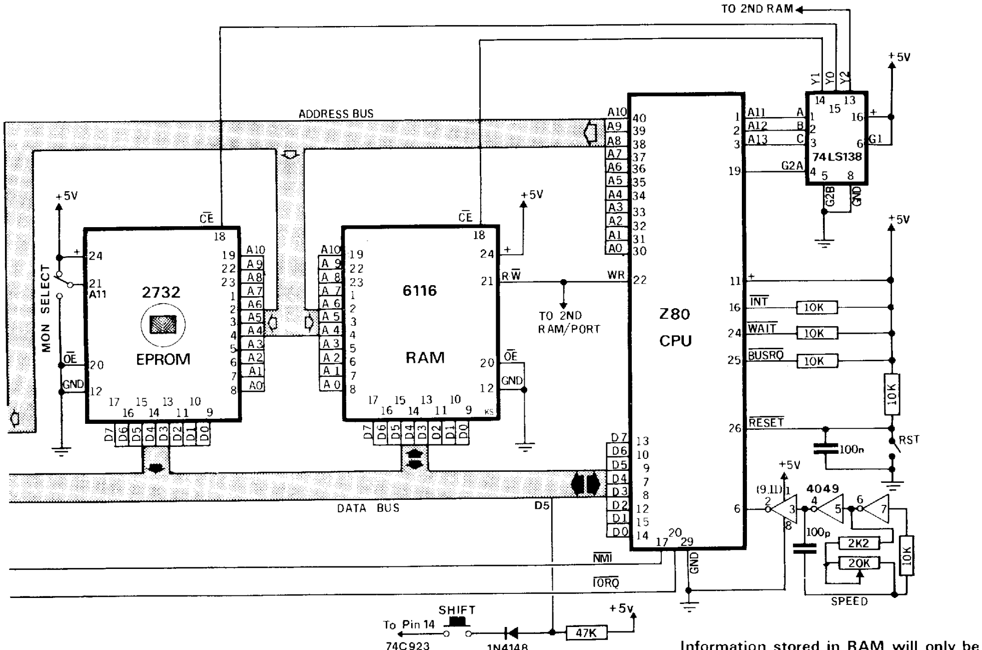
The shaded paths of the diagram represent buses and the address bus

These command lines are called chip select, chip enable or output enable and this allows only one chip to be activated at a time.

The chip select lines are the outputs of a decoder chip and this chip is 'turned on' by the Z-80 and only one of its outputs goes low at a time.

It is a 3-line to 8-line decoder and this means it has 3 input lines and depending on the HIGH-LOW values on these lines, one of the outputs will go low.

This is a form of expander so that a single line from the Z-80 (e.g. from pin 19 or 20) can control 8 devices.



The top right-hand decoder is called the ROM/RAM decoder and the lower left-hand, the IN/OUT decoder.

The data from the monitor flows to the Central Processing Unit (the Z-80) along the data bus as 8 parallel bits of information AT THE SAME TIME.

This is called a BYTE of information and can have 256 different possibilities. The Z-80 knows if the byte is data or instruction by the fact that it starts at address zero looking for an instruction byte. From there the program must follow correctly and this is the responsibility of the programmer.

The data enters the Z-80 via a holding register (an instruction register) that is not available to the programmer and to keep the discussion simple, we consider the byte flows directly into the A register (called the accumulator). This is the only register capable of accepting information from the data bus. All other registers must be fed from the accumulator.

Data can also flow out of the Z-80 along the data bus and this bus is BI-DIRECTIONAL. The arrows on the bus show the direction of flow of information.

The keyboard is scanned by the 74C923 and this is called hardware scanning as the chip has inbuilt scanning circuits for a matrix of 20 keys.

When a key is pressed, a signal is generated at the Data Available pin and the Z-80 is notified via the Non-Maskable Interrupt line.

The Z-80 immediately ceases all processing and jumps to address 66 in the MONitor. Here it executes a short program and activates the input/output decoder to turn on the keyboard encoder. The encoder puts a 5-bit number on the data bus and this is stored for later use or operated upon, as required.

When the shift button is pressed, and kept pressed while one of the keys is pressed, an extra bit is added to create a 6-bit number and thus an additional set of 20 commands can be created.

The output latches are also controlled by the in/out decoder and the control line on each latch is called CP (clock pulse).

When these lines are taken LOW, then HIGH again, the data appearing on the input lines is latched into the chip and will appear on the output lines and will remain there.

This allows devices such as 7-segment displays, relays or globes etc. to be activated.

The 6116 RAM is RANDOM ACCESS MEMORY and as the name suggests, bytes of information can be placed anywhere in its matrix of cells. These bytes are generally data however programs can be stored and run in RAM and these are usually developmental programs.

Information stored in RAM will only be maintained as long as the power is applied as the flip flops storing the data will not hold their state when power is removed.

'ADD-ONs'

This computer is only a baby in the computer world however it does have the facility for expansion and already a number of 'add-ons' have been produced.

Possibly the most important add-on is the NON-Volatile RAM. This consists of a battery backed-up 6116, into which programs can be placed.

Other devices can be connected to the system via the expansion port and this includes an IN/OUT module, an OUTPUT module, a display module and a controller module (to come).

The clock oscillator is adjustable via a speed control pot and allows programs to be run at different speeds for assessment. If a real-time situation is required, a crystal oscillator can be fitted and this will allow time to be programmed accurately.

The main intention of this computer is to provide the starting point for an understanding into computer operations. For this reason, machine code programming has been employed. This means you will be able to create your own systems for such applications as controllers and timers for industry and home and be able to produce the project from the ground up, without requiring any external operating system.

PROGRAMS FOR THE TEC DISPLAYS and a sound Program:

Here are three programs for the TEC and TEC displays. The effects that can be produced on a set of 7-segment displays is quite amazing. I thought we had run out of ideas and yet they still keep coming.

The first program is a Space Invaders sound effect using button 4 as the firing button. The other two programs use the displays.

SPACE INVADERS 'SHOOTING'

Philip Barns. 2118

Computer sounds and effects are always impressive, especially when we have control over them.

This program does just that.

It is a Space Invaders sound effect and you can control it via button 4.

The point to note with this program is the way the delay is increased by inserting a varying value into a delay loop. In the latter half of the program the OFF time is gradually increased by placing another varying value into a delay loop.

The resulting ON-OFF values outputted to the speaker produce the changing tone.

The program only accepts the press of button '4' (determined by **CP 04**) and by pressing this button repeatedly, a firing sound will be produced.

```
LD A,12         800   3E 12
LD I,A          802   ED 47
LD H,FF         804   26 FF
LD B,01         806   06 01
INC B           808   04
LD A,80         809   3E 80
OUT (01),A      80B   D3 01
CALL 0828       80D   CD 28 08
XOR A           810   AF
OUT (01),A      811   D3 02
CALL 0828       813   CD 28 08
LD A,I          816   ED 57
CP 04           818   FE 04
JP Z 0800       81A   CA 00 08
DEC H           81D   25
JP NZ 0808      81E   C2 08 08
CP 04           821   FE 04
JR NZ 0821      823   20 FC
JP 0800         825   C3 00 08

LD C,B          828   48
DEC C           829   0D
JR NZ 0829      82A   20 FD
RETURN          82C   C9
```

THE BOX

G.L. Dunt 3219.

This program is an extension of the techniques we have been discussing in issue 12, P 18, covering the control of two or more pixels at the same time.

It produces an interesting piece of animation in which a box with lid is displayed and moved across the screen in a 'chase scene'.

Again we won't say much about the effect, except to say that you can get quite involved with it and find it very easy to improve upon.

The program consists of 25 'frames' and each frame requires 4 bytes of the table to produce the necessary effects. Each time you increase the table (by 4 bytes) you must also increase the counter register by one (for each frame).

By using 4 bytes we gain the ability to control two pixels at the same time. If only one display is required, the two pairs of bytes will be identical.

```
LD IX 0840        0800   DD 21 40 08
LD D,10           0804   16 19
LD C,40           0806   0E 40
LD A(IX + 00)     0808   DD 7E 00
OUT (01),A        080B   D3 01
LD A(IX + 01)     080D   DD 7E 01
OUT (02),A        0810   D3 02
DJNZ              0812   10 FE
XOR A             0814   AF
OUT (02),A        0815   D3 02
LD A(IX + 02)     0817   DD 7E 02
OUT (01),A        081A   D3 01
LD A(IX + 03)     081C   DD 7E 03
OUT (02),A        081F   D3 02
DJNZ              0821   10 FE
DEC C             0823   0D
JR NZ 0808        0824   20 E2
INC IX            0826   DD 23
INC IX            0828   DD 23
INC IX            082A   DD 23
INC IX            082C   DD 23
DEC D             082E   15
JR NZ 0806        082F   20 D5
JP 0800           0831   C3 00 08
```

at 0840:

```
↓ 01   ↓ 01   ↓ 01   ↓ 20
  E4     E4     80     E4
  01     01     10     20
  E4     E4     C4     E4

  01     01     01     10
  E8     E1     80     E4
  01     01     20     10
  E8     E1     E0     E4

  01     01     02     08
  E4     01     80     E4
  01     02     20     08
  E4     E0     E0     E4

  01     01     04     04
  E2     04     80     E0
  01     02     20     08
  E2     E0     E0     04

  01     01     08     02
  E4     80     80     E0
  01     02     20     08
  E4     E0     E0     04

  01     01     10     01
  E4     80     04     E0
  01     04     20     04
  E4     A4     E0     04

  01     01     20     01
  E2     80     E1     E0
  01     08     20     02
  E2     64     E1     04
```

Halilovic's Piano:

This program has been designed by BOB Halilovic and gives a piano effect when one of the 20 keys is pressed. The notes have a pre-determined length, and this distinguishes it from the organ programs we have previously presented.

```
Data            0800    00
Data            0801    09
LD A,1F         0802    3E 1F
LD (0901),A     0804    32 01 09
CALL 01B0       0807    CD B0 01
HALT            080A    76
CP 10           080B    FE 10
JR NC           080D    30 07
ADD A,05        080F    C6 05
LD (0900),A     0811    32 00 09
JR 0807         08014   18 F1
SUB A,0F        0816    D6 0F
JR 0811         0818    18 F7
```

G. Sheehan & D. Svendsen. 3175

BOOMERANG

Boomerang is a program for the TEC displays. The effect you get is so clever that we are not going to spoil it by telling you what happens.

The only point we will mention is the composition of the byte table.

Each pass of the program uses two bytes from the table and the end of the program is detected by looking for address **0844.** Register L will be 44 at the end of the table.
By using the table two bytes at a time, we can specify the display we wish to access and the segment to be lit.

Also, using a byte table like this requires less program and fewer registers. It is one of the tricks of compact programming.

The delay at 0900 produces the speed of execution.

Try altering and modifying the program and you will learn a lot about what each instruction does. You can also lengthen it by adding more frames. It'll be like creating your own cartoon.

```
LD HL,0820      0800   21 20 08
LD A(HL)        0803   7E
OUT (01),A      0804   D3 01
INC HL          0806   23
LD A,(HL)       0807   7E
OUT (02),A      0808   D3 02
INC HL          080A   23
CALL 0900       080B   CD 00 09
LD A,L          080E   7D
CP 44           080F   FE 44
JP NZ 0803      0811   C2 03 08
JP 0800         0814   C3 00 08
```

at 0820:

```
↓ 01   20   20
  09   C0   6F
  02   10   10
  03   A0   EA
  04   08   08
  06   24   A7
  08   04   04
  0C   44   A7
  10   02   02
  09   C0   28
  20   01   01
  03   A0   C7
```

Delay at 0900:

```
900   11 FF 0A
903   1B
904   7B
905   B2
906   C2 03 09
909   C9
```

PROGRAMS FOR THE 8x8 DISPLAY:

The 8x8 has remained a popular 'add-on' and we still get requests for more programs for it. Here are some recent submissions:

If you have written a program equal to these, send it in for inclusion in the next issue:

FAN OUT Mk III

Dean Svendsen 3175.

FAN OUT Mk III produces symmetry on the displays and can be seen by the same byte being outputted to both ports 3 and 4. The end of the table is detected by looking at the value of L and starting again when it equals the address of the end of the table.

```
LD HL 0815      21 15 08
LD A(HL)        7E
OUT (03),A      D3 03
OUT (04),A      D3 04
INC HL          23
CALL 0900       CD 00 09
LD A,L          7D
CP 20           FE 20
JP NZ 0803      C2 03 08
JP 0800         C3 00 08
```

at 0815:

```
18    81
3C    C3
7E    E7
FF    FF
E7    7E
C3    3C
```

```
900    11 FF 0A
903    1B
904    7B
905    B2
906    C2 03 09
909    C9
```

BOUNCING BALL AND ROLLING BALL.

G.L. Dunt, 3219.

This program is an extension and improvement over the Bouncing Ball program in issue 12, P. 26.

If you look at P.26, you will notice the program is fairly long.

This is because it is necessary to specify the start address of the ball, each time it changes direction.

Much of the program is a repetition of similar or nearly similar codes and to reduce its length we need to look at any part(s) that repeat.

At first they may not be obvious but one can be found that starts at the base of a column, up the column, across to the next and down to the base again. The sequence ends with the LED jumping to the start of the next column.

If we repeat this 4 times, the whole of the board will be covered. This will reproduce the effect as described on P. 26 of issue 12. Using the same technique, we can travel across the display and back again, to produce a weaving effect as the LED advances up the display. To complete the travel we need to move the LED from the top right hand corner to the lower left hand corner, ready for the start of the next sequence.

By using efficient programming as covered in this program, we can produce twice the effect with about half the program.

Most of the reduction is done by defining the co-ordinates of the ball only once. This is done at the beginning of the program and from there the ball position is kept in the C and D registers. They act as the x and y values in co-ordinate geometry.

To move the LED across or up and down the screen, the C and D registers are rotated left or right. Each register contains only one bit and when this moves out the end of the register, it either "sits in the carry box" or passes it and enters the other end of the register. In either case the carry flag is affected and we look for this to let us know the end of the display has been reached.

As you can see, the LED is either "off the end of the board" or at the other side of the display, when the carry is detected and we must shift it back one location, ready for the next run. This way the LED appears to be darting back and forth from one side to the other, and we are not aware of the 'corrections' that take place.

```
LD C,01         0800    0E 01
LD D,01         0802    16 01
LD A,C          0804    79
OUT (03),A      0805    D3 03
LD A,D          0807    7A
OUT (04),A      0808    D3 04
CALL 0900       080A    CD 00 09
RLC D           080D    CB 02
JR NC 0807      080F    30 F6
RR D            0811    CB 1A
RLC C           0813    CB 01
LD A,C          0815    79
OUT (03),A      0816    D3 03
LD A,D          0818    7A
OUT (04),A      0819    D3 04
CALL 0900       081B    CD 00 09
RR D            081E    CB 1A
JR NC,0818      0820    30 F6
RL D            0822    CB 12
RLC C           0824    CB 01
JR NC,0804      0826    30 DC
RRC C           0828    CB 09
LD A,D          082A    7A
OUT (04),A      082B    D3 04
LD A,C          082D    79
OUT (03),A      082E    D3 03
CALL 0900       0830    CD 00 09
RRC C           0833    CB 09
JR NC,082D      0835    30 F6
RL C            0837    CB 11
RLC D           0839    CB 02
LD A,D          083B    7A
OUT (04),A      083C    D3 04
LD A,C          083E    79
OUT (03),A      083F    D3 03
CALL 0900       0841    CD 00 09
RLC C           0844    CB 01
JR NC,083E      0846    30 F6
RRC C           0848    CB 09
RLC D           084A    CB 02
JR NC,082A      084C    30 DC
RRC D           084E    CB 0A
RRC D           0850    CB 0A
LD A,D          0852    7A
OUT (04),A      0853    D3 04
CALL 0900       0855    CD 00 09
RR D            0858    CB 1A
JR NC,0852      085A    30 F6
RRC C           085C    CB 09
LD A,C          085E    79
OUT (03),A      085F    D3 03
CALL 0900       0861    CD 00 09
RRC C           0864    CB 09
JR NC,085E      0866    30 F6
JP 0800         0868    C3 00 08
```

At 0900:

```
LD HL,06FF      21 FF 06
DEC HL          2B
LD A,L          7D
OR H            B4
JP NZ 0903      C2 03 09
Return          C9
```

RAIN DROPS:

Jim Robertson.

This program produces a very effective pattern, similar to falling rain. The random number generator is the interesting part as it is very difficult to produce random numbers in a program that loops.

```
CALL Random Nos.        CD 00 0A
AND 07                  E6 07
LD H,0B         0805    26 0B
LD L,A          0807    6F
RLC (HL)        0808    CB 0E
LD DE,0006      080A    11 06 00
CALL SCAN       080D    CD 00 09
DEC DE          0810    1B
LD A,D          0811    7A
OR E            0812    B3
JRNZ            0813    20 F8
JR START        0815    18 E9
```

at 0900:

SCAN

```
LD HL 0B00      0900    21 00 0B
LD B,01         0903    06 01
LD A(HL)        0905    7E
OUT (03),A      0906    D3 03
LD A,B          0908    78
OUT (04),A      0909    D3 04
LD B,20         090B    06 20
DJNZ            090D    10 FE
INC HL          090F    23
LD B,A          0910    47
XOR A           0911    AF
OUT (04),A      0912    D3 04
RLC B           0914    CB 00
JR NC           0916    30 ED
RETURN          0918    C9
```

at 0A00:

RANDOM NUMBERS:

```
LD A,R          0A00    ED 5F
LD B,A          0A02    47
LD A,R          0A3     ED 5F
RLA             0A05    17
LD R,A          0A06    ED 4F
DJNZ            0A08    10 FB
RETURN          0A0A    C9
```

PHONE DIALLER

TURNING THE TEC INTO A PHONE DIALLER

The following three or four pages examine the development of an idea. It is a Telephone Dialler capable of storing up to 30 or 40 names and phone numbers with a dialling facility and auto re-dial.

It is only a program of ideas as the output appears on a speaker in the form of tones.

Since this is a fiarly ambitious concept, it has been divided into 3 sections. Each section describes a program that is complete in itself and increases in complexity with complete design in section 3.

The first program is fairly simple. It shows how to get figures from the keyboard and display them on the screen. The second contains two function buttons, **C** and **E**. The 'C' key clears the screen and 'E' indicates the end of a phone number.
The third program is much more complex. It has more features and is keeping track of more things.

Each program has been created from scratch as it is almost impossible to 'add onto' an existing program.

Type each of these programs into the TEC and study them. This way you will learn how they operate.

PHONE DIALLER PROGRAM 1.

This program is limited to displaying 6 digits on the TEC screen as no scrolling feature is present. As the keys are pressed, the numbers fill the screen from left to right. When the screen is full, the capability of the program is reached.

The screen buffer is located at **0900** and the scan rate is determined by the value of B (at **082E** and **082F**). We can increase or reduce the scan rate by altering the value of B and by adjusting the TEC clock speed.

No other features are available in this program. The TEC must be reset and 'GO' pushed to clear the screen so that a new number can be keyed in.

This simple program shows how to get numbers from the keyboard and onto the screen.

The only instruction that will be unfamiliar is **JRNC.** It effectively divides the keyboard in two, allowing keys 0-9 to be accepted and A-F to be disregarded.

JRNC means Jump Relative if the Carry flag is NOT SET. When the previous instruction is a 'COMPARE', it is best to substitute the word 'BORROW' for carry, and the instruction will be much easier to understand. This is because the compare instruction subtracts the data byte from the accumulator and if a borrow is required, the carry flag is SET.

PHONE DIALLER - Part 1

LD D, 08	0800	16 08	The first 8 memory locations are cleared so that the program will come on with a blank screen. We need only 6 locations. The 7th location is explained in the text. Register A is zeroed and this value is inserted into **0900 - 0907** via the HL register being the pointer register.
XOR A	0802	AF	
LD HL,0900	0803	21 00 09	
LD (HL),A	0806	77	
INC HL	0807	23	
DEC D	0808	15	
JR NZ	0809	20 FB	
LD A,I	080B	ED 57	The Index register contains the value of the key.
CP 0A	080D	FE 0A	Compare the accumulator with **0A.**
JR NC	080F	30 12	Jump relative if the key is A or higher.
LD DE 0880	0811	11 80 08	Load DE with the start of the DISPLAY TABLE.
ADD A,E	0814	83	Add **80** to the key value.
LD E,A	0815	5F	Load the result back into E. DE will point to a table-byte.
LD HL,0900	0816	21 00 09	Load HL with the start of memory.
LD A,(HL)	0819	7E	Look for the first blank memory location by loading the value pointed to by HL into the accumulator and comparing with zero until a blank location is found.
CP 00	081A	FE 00	
JR Z	081C	28 03	
INC HL	081E	23	
JR	081F	18 F8	
LD A(DE)	0821	1A	When found, load A with the byte pointed to by DE.
LD (HL),A	0822	77	Load the table value into the blank memory location.
LD A,FF	0823	3E FF	Change the value of the index register by loading it with FF so that we can detect the same or another button.
LD I,A	0825	ED 47	
LD C,20	0827	0E 20	start the scan at the left hand end of the display.
LD HL,0900	0829	21 00 09	Load HL with start of memory.
LD D,06	082C	16 06	Load D with 06 for 6 loops of the program.
LD B,00	082E	06 00	Load B with delay value for turning ON each digit.
LD A(HL)	0830	7E	Load the data at the first memory location into A.
OUT (02),A	0831	D3 02	Output to the segment port.
LD A,C	0833	79	Load C into A.
OUT (01),A	0834	D3 01	Output to the cathode port.
RRC C	0836	CB 09	Rotate register C right, to access the 2nd display.
DJNZ	0838	10 FE	Create a short delay to display the digit.
XOR A	083A	AF	Zero A
OUT (01),A	083B	D3 01	Output to the cathode port to turn display OFF.
INC HL	083D	23	Increment to the next location.
DEC D	083E	15	Decrement the loop register.
JR NZ	083F	20 ED	Jump to start of loop if D not zero.
JR	0841	C3 0B 08	Jump to start of program if D zero and look for new key.

In our program, **CP 0A** causes the Z-80 to substact **0A** from the accumulator (it will hold the value of the key). When any key below **A** is pressed, the subtraction operation creates a borrow and this sets the carry flag. If we push key **6**, the operation will be 6 - A and the answer will require a borrow. Thus the carry flag will be SET. If we go to the program, we can see the Z-80 will continue down the program and NOT JUMP as the instruction says: JUMP RELATIVE NO BORROW.

To fully understand these instructions you have to comprehend the double negative. For instance: I am NOT, NOT going to jump means I AM going to jump.

Type the program at **0800** and the display conversion table at **0880.**

Push RESET, GO and the displays will blank. Press any combination of keys and notice that only number keys respond.

Modify the value of B in the scan section to increase the scan rate.

Some ideas for experimenting include: scanning from the opposite direction, scanning only 5 displays, allowing letters to appear on the screen, and changing the output to a CODE, so that you can turn it into a CODE-BREAKING game.

at 0880:

EB
28
CD
AD
2E
A7
E7
29
EF
AF

PHONE DIALLER - Part 2

The second part of the Phone Dialler program uses a different approach. As we have said, each must start afresh as it is more difficult to adapt an existing program.

This program accepts a string of digits of any length and will remember them for recall after key **E** (for END) has been pressed.

The **C** button clears the display and can be pressed at any time. When the desired number has been entered, button E is pressed. The display is blanked and the numbers emerge from the right hand end of the display and shift across to the left. Three empty spaces are created before the numbers start again.

This program introduces the concept of control keys and also the need for sub-routines for any sequence that is required more than once.

Programs increase in length as more and more housekeeping is called for. Housekeeping is looking for button presses or detecting the end of a sequence etc.

The prime requirement of the program is to keep the displays illuminated. This means we must be calling SCAN for most of the time and as you will see, the SCAN routine is a favourite place to put house-keeping.

If you want a key to be immediately responsive, it must be checked during the SCAN loop. To be more precise, it must be checked during the inner-most loop as this is the loop which is being run for most of the time.

Key the program into the TEC and run it. Try changing some of the locations and see the result. This is the best way to following what is happening, especially at specific locations.

HOW THE PROGRAM WORKS

The program generates 2 memory areas. One is made up of 6 locations, from **0900** to **0905** and is called the **DISPLAY BUFFER.** The other is from **0907** onwards and is called **MEMORY AREA.**

The SCAN ROUTINE (at 0877) looks at the Display Buffer locations and outputs their value onto the displays.

The remainder of memory, starting at 0907 holds any number of digital as required and is open-ended.

One location, **0906**, is left blank and its purpose will be explained later.

As each number is keyed in, it is stored in memory, from 0907 onwards, and the HL register pair keeps track of the next available location.

The number is also outputted onto the display but firstly a SHIFT ROUTINE is called. The function of this routine is to take the value corresponding to the left-hand digit and drop it out of the buffer zone. The second location is then transferred to the first, the third to the second etc until all the digits have been shifted one place to the left. This leaves an empty hole at the right-hand end of the display.

The way in which this empty space is generated is quite clever. The **'00'** in **0906** is shifted into the 6th buffer location.

The program then loads the present key value in the buffer zone, position six, and reverts to a scan situation in which it is looking for an 'end of number' via button E.

When this is detected, memory is incremented one location and E is inserted.

The displays are cleared and the program picks up the first digit at 0907and places it in the 6th position of the buffer area.

The shift routine is called then the next memory value is placed in the 6th buffer location.

Before each new value is loaded into the buffer area, it is compared with **0E** to detect the 'end of message.'

When E is detected, three blank locations are produced and the message starts again.

The CLEAR function is included in the SCAN routine. This has been done so that CLEAR can be detected instantly, as the display scan must be running at all times to keep the displays illuminated.

DIALLER Part 2 listing:
Main Program:

```
LD D,20          0800  16 20
CALL CLEAR       0802  CD 5B 08
LD HL, 0907      0805  21 07 09
LD A,I           0808  ED 57
CP 0A            080A  FE 0A
JR NC,0820       080C  30 12
INC HL           080E  23
LD DE,08A5       080F  11 A5 08
ADD A,E          0812  83
LD E,A           0813  5F
CALL SHIFT       0814  CD 65 08
LD A,(DE)        0817  1A
LD (HL),A        0818  77
LD (0905),A      0819  32 05 09
LD A,FF          081C  3E FF
LD I,A           081E  ED 47
CP 0E            0820  FE 0E
LR Z,002A        0822  28 05
CALL SCAN        0824  CD 77 08
JR 0808          0827  18 DF
INC HL           0829  23
LD (HL),A        082A  77
LD D,06          082B  16 06
CALL CLEAR       082D  CD 5B 08
LD HL,0907       0830  21 07 09
LD A,(HL)        0833  7E
LD D,20          0834  16 20
INC HL           0836  23
CP 0E            0837  FE 0E
JR Z,0849        0839  28 0E
LD (0905),A      083B  32 05 09
CALL SCAN        083E  CD 77 08
DEC D            0841  15
JR NZ,083E       0842  20 FA
CALL SHIFT       0844  CD 65 08
JR 0833          0847  18 EA
LD E,02          0849  1E 02
LD D,20          084B  16 20
CALL SCAN        084D  CD 77 08
DEC D            0850  15
JR NZ,084D       0851  20 FA
CALL SHIFT       0853  CD 65 08
DEC E            0856  1D
JR NZ,084B       0857  20 F2
JR 0830          0859  18 D5
```

Clear:

```
XOR A            085B  AF
LD HL,0900       085C  21 00 09
LD (HL),A        085F  77
INC HL           0860  23
DEC D            0861  15
JR NZ, 085F      0862  20 FB
RETURN           0864  C9
```

Shift:

```
LD B,07            0865  06 07
LD IX,08FF         0867  DD 21 FF 08
LD A,(IX + 01)     086B  DD 7E 01
LD (IX + 00),A     086E  DD 77 00
INC IX             0871  DD 23
DEC B              0873  05
JR NZ,086B         0874  20 F5
RETURN             0876  C9
```

Scan:

```
PUSH HL          0877  E5
PUSH DE          0878  D5
LD C,20          0879  0E 20
LD HL,0900       087B  21 00 09
LD D,06          087E  16 06
LD B,80          0880  06 80
LD A,(HL)        0882  7E
OUT (02),A       0883  D3 02
LD A,C           0885  79
OUT (01),A       0886  D3 01
RRC C            0888  CB 09
DJNZ 088A        088A  10 FE
XOR A            088C  AF
OUT (01),A       088D  D3 01
INC HL           088F  23
LD A,I           0890  ED 57
CP 0C            0892  FE 0C
JR Z,089C        0894  28 06
DEC D            0896  15
JR NZ,0880       0897  20 E7
POP DE           0899  D1
POP HL           089A  E1
RETURN           089B  C9
POP DE           089C  D1
POP HL           089D  E1
LD A,FF          089E  3E FF
LD I,A           08A0  ED 47
JP 0800          08A2  C3 00 08
```

at 08A5:

```
0 = EB
1 = 28
2 = CD
3 = AD
4 = 2E
5 = A7
6 = E7
7 = 29
8 = EF
9 = AF
0 =
```

PHONE DIALLER - Part 3

The third and final part of the Phone Dialler program is the longest and most impressive. It looks complicated because it is looking after a lot of things.

The program accesses memory and when using the 2k onboard RAM, it is capable of holding up to 36 names and numbers, each fitting into a block of memory 20H bytes long. The program allows up to 27 characters for the name and number and this should be sufficient for any situation.

The program uses a lot of sub-routines and they perform most of the work.

As the processor goes through the MAIN program, it CALLS the sub-routines and they do all the displaying, shifting, display converting etc.

Any operation that is required more than once is put into the form of a sub-routine. This reduces the length of the program and allows the sub-routines to be called as many times as required.

USING THE PROGRAM

Basically the program is self explanatory as the instructions for its use are displayed on the screen after the GO button is pressed.

The first instruction is to select an INDEX NUMBER from 00 to 36 (decimal) into which the telephone number is placed.

Push button E and the screen will blank so that the index number can be inserted.

The index number will remain on the screen for about one second and then the second set of instructions will appear. After reading the instructions, push E. This will cause the screen to blank so that you can type the name corresponding to the phone number.

After the end of the name, insert a space by typing F and the program will convert to displaying a digit for each key pressed.

At the end of the phone number type E and the program will scroll the contents of memory.

To dial the phone number push D. The program will pause for 5 seconds then dial the number.

At the completion of dialling, the screen will scroll the name and number again.

You can redial the same number at any time by pressing D.

To re-load the memory BLOCK, push C. This will re-start the program and allow a new name and number to be inserted.

Once a name and number has been inserted into memory at a particular index value, it can be dialled very quickly. You can push either button C or RESET. If the Reset button is pushed, the GO button must be pushed for the first set of instructions to appear.

Push E and insert the index number; then push D. The computer will dial the number. A constant beeping will indicate the location is not filled and you should try another index.

At the end of dialling, the name and number will scroll and you can confirm it to be correct.

A SUMMARY OF THE PROGRAM

The program creates a display buffer area at **0A80** to **0A85** and the values placed at these 6 locations are directly transferred to the TEC display via the SCAN routine.

The CLEAR routine zeros each of these locations and also the next location. This is one of the clever tricks of the program, and it is cleared for the following reason:

The SHIFT routine starts at a location that is one lower than **OA80**, (namely **OA7F**) and places the data at **OA80** into

PHONE DIALLER PROGRAM:

CALL CLEAR	0800	CD 20 09	The first 7 lines of the program displays "Enter Index .
LD HL,0A0C	0803	21 0C 0A	. . . . etc and looks for the value **10** at the end of the
CALL SCROLL	0806	CD C0 09	table to repeat the sequence. The program also looks
CP 10	0809	FE 10	for an input value above 9 to jump out of the loop.
JR Z,0803	080B	28 F6	
CP 0A	080D	FE 0A	
JR C,0800	080F	38 EF	
CALL CLEAR	0811	CD 20 09	The screen is cleared and the index register is loaded
LD A,FF	0814	3E FF	with FF so that we can detect when a button has been
LD I,A	0816	ED 47	pushed.
LD HL,0000	0818	21 00 00	Memory is set to zero by loading HL with **00 00.**
LD A,01	081B	3E 01	Location **09FE** stores the value 01 so that key value is
LD (09FE),A	081D	32 FE 09	called once. The requirement of the next 12 lines is to
CALL KEY VALUE		CD 30 09	get a double decimal number into location **09FC.**
LD A,C	0823	79	C will contain the key value and this is loaded into
LD (09FC),A	0824	32 FC 09	memory location **09FC** (first figure).
LD A,01	0827	3E 01	Repeat the sequence and call KEY VALUE once more.
LD (09FE),A	0829	32 FE 09	
CALL KEY VALUE	082C	CD 30 09	
LD A,(09FC)	082F	3A FC 09	Load the first figure into A and rotate the accumulator
RLA	0832	17	4 places to the left to shift the number into the upper
RLA	0833	17	half of the register.
RLA	0834	17	
RLA	0835	17	
ADD A,C	0836	81	Add the second figure to the accumulator and store
LD (09FC),A	0837	32 FC 09	the result into **09FC** as a two figure decimal number.
LD D,20	083A	16 20	Create a delay with register D and call SCAN for 20H
CALL SCAN	093C	CD 80 09	loops. (32 loops).
DEC D	083F	15	
JR NZ,083C	0840	20 FA	
CALL CLEAR	0842	CD 20 09	Clear the display and load the pointer register with the
LD HL,0A2C	0845	21 2C 0A	start address of the second table. Display "Enter
CALL SCROLL	0848	CD C0 09	name . . . . .etc" Look for the end of the table (**10**) and
LD A,(HL)	084D	7E	loop, unless a key 0-9 has been pressed.
CP 10	084C	FE 10	
JR Z,0845	084E	28 F5	
CP 0A	0850	FE 0A	
JR C,0842	0852	38 EE	
CALL CLEAR	0854	CD 20 09	Call CLEAR to clear the display.
CALL MEM ADDR	0857	CD 60 09	Read MEMORY ADDRESS notes.
LD D,1C	085A	16 1C	Register D counts up to 28 characters (max allowed).
LD E,00	085C	1E 00	Register E counts to 2. Two key presses for a char.
LD A,FF	095E	3E FF	Fill the I register via the accumulator so that we can
LD I,A	0860	ED 47	detect when a key is pressed.
CALL SCAN 2	0862	CD D0 0A	Scan the display looking for a key press 0-F.
LD A,I	0865	ED 57	
CP 10	0867	FE 10	
JR NC,0862	0869	30 F7	
INC E	086B	1C	Increment the E register.
LD A,E	086C	7B	Load E into A.
CP 02	086D	FE 02	Compare the accumulator with 02 and jump if the two
JR Z,087C	086F	28 0B	are the same. If not, go to the next instruction.
LD A,I	0871	ED 57	Look to see if a space is required as this will indicate
CP 0F	0873	FE 0F	the end of names and the beginning of numbers.
JR Z,0895	0875	28 1E	Jump relative if F has been pressed.
LD (09FA),A	0877	32 FA 09	Store the value of A at **09FA** and loop for second press
JR 085E	087A	18 E2	of button.
CALL SHIFT	087C	CD E1 09	Call SHIFT to get display ready for next number.
LD A,(09FA)	087F	3A FA 09	Load the first number into the accumulator and shift it
RLA	0882	17	4 places to the left to occupy the upper half of the
RLA	0883	17	register.
RLA	0884	17	
RLA	0885	17	
LD B,A	0886	47	Save the result in B.
LD A,I	0887	ED 57	Put second number into the accumulator.
ADD A,B	0889	80	Combine the two to create a 2-digit number.
LD (HL),A	088A	77	Load this value into the location looked at by HL.
LD (0A85),A	088B	32 85 0A	Also load it into the first display location.
INC HL	088E	23	Increment HL.
DEC D	088F	15	Decrement D and
JR NZ,085C	0890	20 CA	Jump if **1C** locations not filled.
JP 0800	0892	C3 00 08	Jump to start if overflow occurs.
XOR A	0895	AF	Zero A and load it
LD (HL),A	0896	77	into the location looked at by HL to create a space.
CALL SHIFT	0897	CD E1 09	Shift the display digits one place to the left .
LD A,D	089A	7A	Load the remaining locations into A and store at **09FE**
LD (09FE),A	089B	32 FE 09	for use by the CALL KEY routine.
CALL KEY VALUE	089E	CD 30 09	Call KEY VALUE. This will put Nos onto the display.
LD B,03	08A1	06 03	Create 3 blank locations after te numbers have been
INC HL	08A3	23	inserted, to produce a space between the end of the
XOR A	08A4	AF	message and the start so that it can be scrolled across
LD (HL),A	08A5	77	the display.
DEC B	08A6	05	
JR NZ,08A3	08A7	20 FA	
INC HL	08A9	23	
LD A,10	08AA	3E 10	Increment HL and load last location with **10** so that
LD (HL),A	08AC	77	program will loop name and telephone number.
NOP	08AD	00	

this lower location. As can be seen from the program, this lower location is not displayed on the TEC and thus the data shifts off the screen. The data for the second location is shifted to the location for the first display and this repeats for the 6 locations. The result is the data in the blank location at **0A86** is shifted into the last display location and thus an empty space is produced on the display.

It is important for **0A86** to be empty for this to work.

The MEMORY ADDRESS routine creates areas that are 20H bytes long and starts at **0B00.**

The program stores the Index number at location **09FC** and as each memory area is created, it decrements the Index number and the program exits when the count register is zero.

The HL register will contain the start of this address. It is not used for any other purpose and thus it will not be destroyed during the running of the program and will hold the current value for re-dial, if required.

The SCROLL routine picks up the first byte from the table and places it at **0A85** and then calls SCAN for 20H loops (32 passes of the display).

The SHIFT routine is then called and all the bytes (including the blank locations) are transferred one position to the left.

The scroll program then loops and repeats the sequence until the end of the table is reached. It detects this by looking for 10H (we could have chosen any value) and the message re-starts.

When the 'Dial key' '**D**' is pressed, a BEEP routine and PAUSE routine are called. These produce a suitable ON-OFF tone to the speaker and the program converts the values in memory to a string of beeps.

The program ignores the name at the beginning of memory and looks for the first location containing zero.

The end of the phone number is detected by also looking for a location containing zero.

The program then jumps back to calling the start of memory and scrolls the message across the screen.

SUGGESTIONS

The program can be keyed into the TEC and fills about 3 pages, from **0800** to **0AEE.**

After this is done, it is wise to save a copy of the program in non-volatile RAM so that it is not lost.

To save the program, type the following dump routine at **0F80:**

```
11 00 10
21 00 08
01 90 07
ED B0
C7
```

CALL CLEAR	08AE	CD 20 09	Clear the screen.
CALL MEM ADDR	08B1	CD 60 09	Get start of BLOCK via **09FC** (36 blocks available).
CALL SCROLL	08B4	CD C0 09	Scroll name and number across screen.
CP 10	08B7	FE 10	Look for end of message. If another key is pressed,
JR Z,08B1	08B9	28 F6	jump out of loop.
LD B,20	08BB	06 20	Create a pause before dialling by loading B with 20
CALL PAUSE	08BD	CD 72 09	and calling pause 32 times. This creates approx 2
DJNZ 08BD	08C0	10 FB	second delay.
CALL CLEAR	08C2	CD 20 09	Clear the screen of any junk etc.
CALL MEM ADDR	08C5	CD 60 09	Get start of block (00-36).
LD A,(HL)	08C8	7E	Look for space between name and phone number by
INC HL	08C9	23	comparing the contents of each location with 00 and
CP 00	08CA	FE 00	incrementing until 00 is found.
JR NZ,08C8	08CC	20 FA	The next 6 lines create the dialling pulses by loading
LD IX,0A00	08CE	DD 21 00 0A	IX with the start of the number table and calling BEEP
INC IX	08D2	DD 23	routine. (The beep calls a pause). The program then
CALL BEEP	08D4	CD 00 09	compares the byte in the table with the byte in the
LD A,(IX + 00)	08D7	DD 7E 00	block and loops until a comparison is found. Note: we
CP (HL)	08DA	BE	go into the routine 'blind' and beep before a CP!!
JR NZ,08D2	08DB	20 F5	Create a short pause at the end of each digit so that
LD B,10	08DD	06 10	the phone system detects the end of a digit.
CALL PAUSE	08DF	CD 72 09	Increment to next digit, look to see if end of phone
DJNZ 08DF	08E2	10 FB	number has been reached and return to above routine
INC HL	08E4	23	for next set of pulses.
LD A,(HL)	08E5	7E	
CP 00	08E6	FE 00	
JR Z,08EC	08E8	28 02	
JR 08CE	08EA	18 E2	
LD A,I	08EC	ED 57	If no buttons have been pressed during dialling, I will
CP 0D	08EE	FE 0D	still contain 0D (from above) and program will scroll
JR Z,08AE	08F0	28 BC	name and number. If any other key has been pressed,
JR 08EC	08F2	18 F8	program will loop with blank screen until D pressed.

This is the end of the MAIN PROGRAM. The sub-routines below are called by the main program.

BEEP

PUSH AF	0900	F5	Registers A, B and C are used in this sub-routine and
PUSH BC	0901	C5	thus they must be pused onto the stack and saved.
LD B,20	0902	06 20	Reg B holds the number of cycles for the beep routine
LD A,80	0904	3E 80	Register A turns on the speaker bit.
LD C,20	0906	0E 20	Reg C holds the turn-on cycles for the spkr.
OUT (01),A	0908	D3 01	The spkr is turned on via **OUT (01),A**
DEC C	090A	0D	and a delay created via register C for
JR NZ,090A	090B	20 FD	32 loops.
LD C,20	090D	0E 20	The same OFF delay period is created via register C
XOR A	090F	AF	for an even 'mark-space' ratio for the speaker.
OUT (01),A	0910	D3 01	
DEC C	0912	0D	
JR NZ,0912	0913	20 FD	
DEC B	0915	05	The count register (register B) is decremented and the
JR NZ,0904	0916	20 EC	program loops until B is zero.
CALL PAUSE	0918	CD 72 09	The program calls pause to produce silence.
POP BC	091B	C1	Registers A, B and C are popped off the stack and will
POP AF	091C	F1	contain the original values and before the routine.
RETURN	091D	C9	Return to the main program.

CLEAR

LD D,07	0920	16 07	This routine clears the 6 display locations **0A80** to
XOR A	0922	AF	**0A85** and also **0A86** by zeroing A and
LD HL,0A80	0923	21 80 0A	loading HL with start address of buffer zone
LD (HL),A	0926	77	and loading zero into the location pointed to by HL.
INC HL	0927	23	INC HL
DEC D	0928	15	DEC D
JR NZ,0926	0929	20 FB	and jump for 7 loops.
RETURN	092B	C9	Return to main program.

KEY VALUE

LD DE,0A00	0930	11 00 0A	Load DE to point to beginning of number table.
LD A,I	0933	ED 57	Load key value into accumulator.
CP 0A	0935	FE 0A	Compare with 0A and jump if the key value is A-F or
JR NC,0952	0937	30 19	not pressed or go to next instruction if 0-9.
INC HL	0939	23	INC HL (used when creating phone number)
LD C,A	093A	4F	Save A in C.
ADD A,E	093B	83	ADD the start of table to A (table may start at 0A03!).
LD E,A	093C	5F	Make DE ready to point at value in table.
CALL SHIFT	093D	CD E1 09	SHIFT display contents one place to left.
LD A,(DE)	0940	1A	Load byte from number table into accumulator.
LD (HL),A	0941	77	Load number byte into loaction in BLOCK.
LD (0A85),A	0942	32 85 0A	and also into right hand display.
LD A,FF	0945	3E FF	Load A with FF and then into I to detect when another
LD I,A	0947	ED 47	key has been pressed.
LD A,(09FE)	0949	3A FE 09	**09FE** caontains 01 via beginning of of main program
DEC A	094C	3D	and KEY VALUE is called once. Or **09FE** contains **1C**
LD (09FE),A	094D	32 FE 09	to keep track on the number of locations being filled in
RET Z	0950	C8	the BLOCK.
XOR A	0951	AF	Zero A.
CP 0E	0952	FE 0E	Compare accumulator with E and RETURN if E key is
RET Z	0954	C8	pushed. Otherwise call SCAN and display the
CALL SCAN	0955	CD 80 09	contents of the 6 memory locations. Jump to stat of
JR 0930	0958	18 D6	KEY VALUE sub-routine and loop until 0-9 pressed.

Decrement to **0F80** and push GO. Make sure the non-volatile RAM switch is on RAM (read/write) so that the data will be accepted. Check that the program has been dumped by addressing **1000** and compare the data with the listing.

If you have inserted names and numbers into index locations and want to save them, address **0F80** and push GO. Make sure the RAM card is in read/write mode and everything will be saved.

Switch to ROM mode and everything will be preserved.

You can now turn the TEC off.

To transfer the program back to **0800**, address **1780** and change 2 of the bytes to the following:

```
11 00 08  <-  these two bytes
21 00 10  <-  are changed
01 90 07
ED B0
C7
```

Decrement to **1780** and push GO. The RAM card should be in ROM MODE for this operation.

Push GO again and the program will run.

All names and numbers will be available.

AUTO REDIAL

An automatic re-dial facility can also be included so that the number automatically re-dials after say 5 or 10 minutes; if the number was originally engaged. This is very handy for those occassions when you particularly want to contact a person and their number is busy. By the time you get around to calling again, they have gone!

A simple addition to the program can be fitted in at **08BE** and this will create a delay by counting the number of times the name and phone number scroll past the display. This is only a suggestion and we have not actually produced the program for re-dial.

Register E is the 'count register' and the remainder of the program remains the same. The only bytes you will have to change are jump relative values as well as the jump value at **09B4**. You may also need a subroutine and a flag to pick up redial mode.

Here is a suggested AUTO RE-DIAL program for insertion at **08B4:**

```
LD E,40
DEC E
JR Z
CALL CLEAR
CALL MEMORY ADDR
CALL SCROLL
CP 10
JR Z
CALL CLEAR

JR
```

MEMORY ADDRESS

```
LD HL,0B00      0960   21 00 0B
LD A,(09FC)     0963   3A FC 09
LD D,20         0966   16 20
CP 00           0968   FE 00
RET Z           096A   C8
INC HL          096B   23
DEC D           096C   15
JR NZ,096B      096D   20 FC
DEC A           096F   3D
JR 0966         0970   18 F4
```

Memory Address sub-routine locates the beginning of the name and phone number block. Each block is 20H bytes long (32 bytes) and memory starts at **0B00.** The BLOCK No is stored at **09FC** and the program increments 20H loops for each block by decrementing register D to zero, then decrementing register A by ONE. This is repeated until A is zero. The sub-routine then exits. HL pair is constantly incremented during this program and will point to the start of the block we want.

PAUSE

```
XOR A           0972   AF
OUT (01),A      0973   D3 01
LD DE,02FF      0975   11 FF 02
DEC DE          0978   1B
LD A,E          0979   7B
OR D            097A   B2
JR NZ,0978      097B   20 FB
RETURN          097D   C9
```

Pause produces a silence from the speaker by outputting zero to port 01. Register DE is decremented and 'wastes computer time' for about 1/10th second. This sub-routine then returns to where it has been called.

SCAN 1

```
PUSH HL         0980   E5
PUSH DE         0981   D5
LD C,20         0982   0E 20
LD HL,0A80      0984   21 80 0A
LD D,06         0987   16 06
LD B,20         0989   06 20
LD A,(HL)       098B   7E
OUT (02),A      098C   D3 02
LD A,C          098E   79
OUT (01),A      098F   D3 01
RRC C           0991   CB 09
DJNZ 0993       0993   10 FE
XOR A           0995   AF
OUT (01),A      0996   D3 01
INC HL          0998   23
LD A,I          0999   ED 57
CP 0C           099B   FE 0C
JR Z,09A9       099D   28 0A
CP 0D           099F   FE 0D
JR Z,09B2       09A1   28 0F
DEC D           09A3   15
JR NZ,0989      09A4   20 E3
POP DE          09A6   D1
POP HL          09A7   E1
RETURN          09A8   C9
POP DE          09A9   D1
POP HL          09AA   E1
LD A,FF         09AB   3E FF
LD I,A          09AD   ED 47
JP 0800         09AF   C3 00 08
POP DE          09B2   D1
POP HL          09B3   E1
JP 08BB         09B4   C3 BB 08
```

The SCAN routine uses H, L and D registers and thus they must be pushed onto the stack and saved.
Load HL with start of display buffer.
The routine displays 6 locations.
The left-hand display is accessed via line '20'.
Load B with a short delay value.
Load the byte at the first location into A.
Output to port 02.
Load C into A, and
output to port 01. This will turn on left-hand display.
Rotate register C to the right for the next display.
Short delay via register B.
Zero A, and
output to port 01.
Look at next memory location.
Load the keyboard value into A.
Look to see if CLEAR has been pressed.
Jump if it has.
DEC D ready for outputting to the next display.
Jump relative if D is not zero.
Pop DE and HL register pairs off the stack.

and RETURN to the main program.
If CLEAR has been pressed, pop DE and HL and load the I register with **FF** so that the program will detect when another key has been pressed.

Jump to **0800.**
POP DE and HL and jump to **08BB** if D (DIALS) has been pressed.

SCAN 2

```
PUSH HL         0AD0   E5
PUSH DE         0AD1   D5
LD C,20         0AD2   0E 20
LD HL,0A80      0AD4   21 80 0A
LD D,06         0AD7   16 06
LD B,20         0AD9   06 20
LD A,(HL)       0ADB   7E
OUT (02),A      0ADC   D3 02
LD A,C          0ADE   79
OUT (01),A      0ADF   D3 01
RRC C           0AE1   CB 09
DJNZ 0AE3       0AE3   10 FE
XOR A           0AE5   AF
OUT (01),A      0AE6   D3 01
INC HL          0AE8   23
DEC D           0AE9   15
JR NZ,0AD9      0AEA   20 ED
POP DE          0AEC   D1
POP HL          0AED   E1
RETURN          0AEE   C9
```

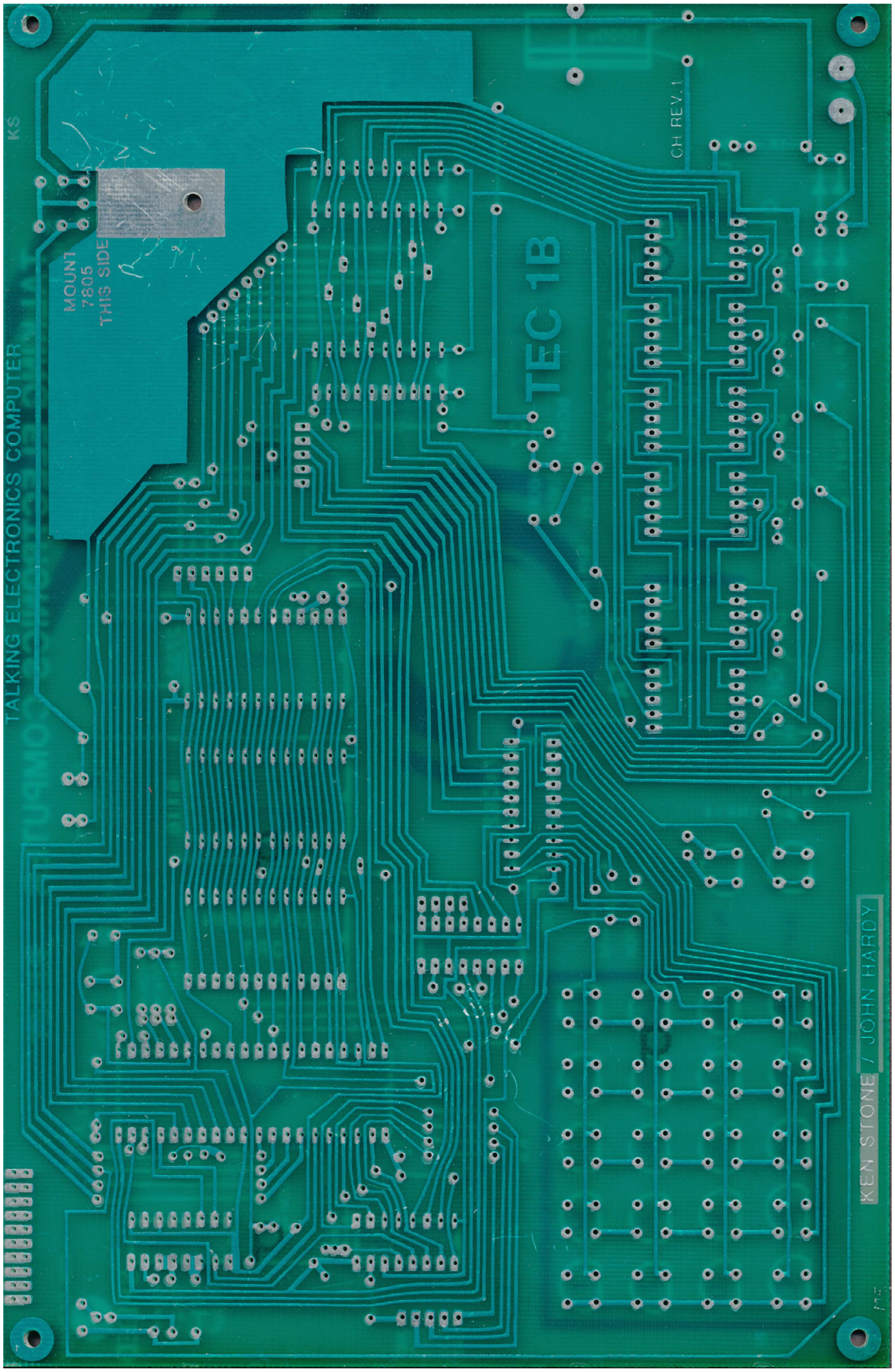
SCAN 2 is identical to SCAN 1 in the scanning section. The only difference is the 'checking' instructions, to see if a particular key is pressed. SCAN 1 above checks to see if a function key is pressed, whereas SCAN 2 performs the scan without any checks.

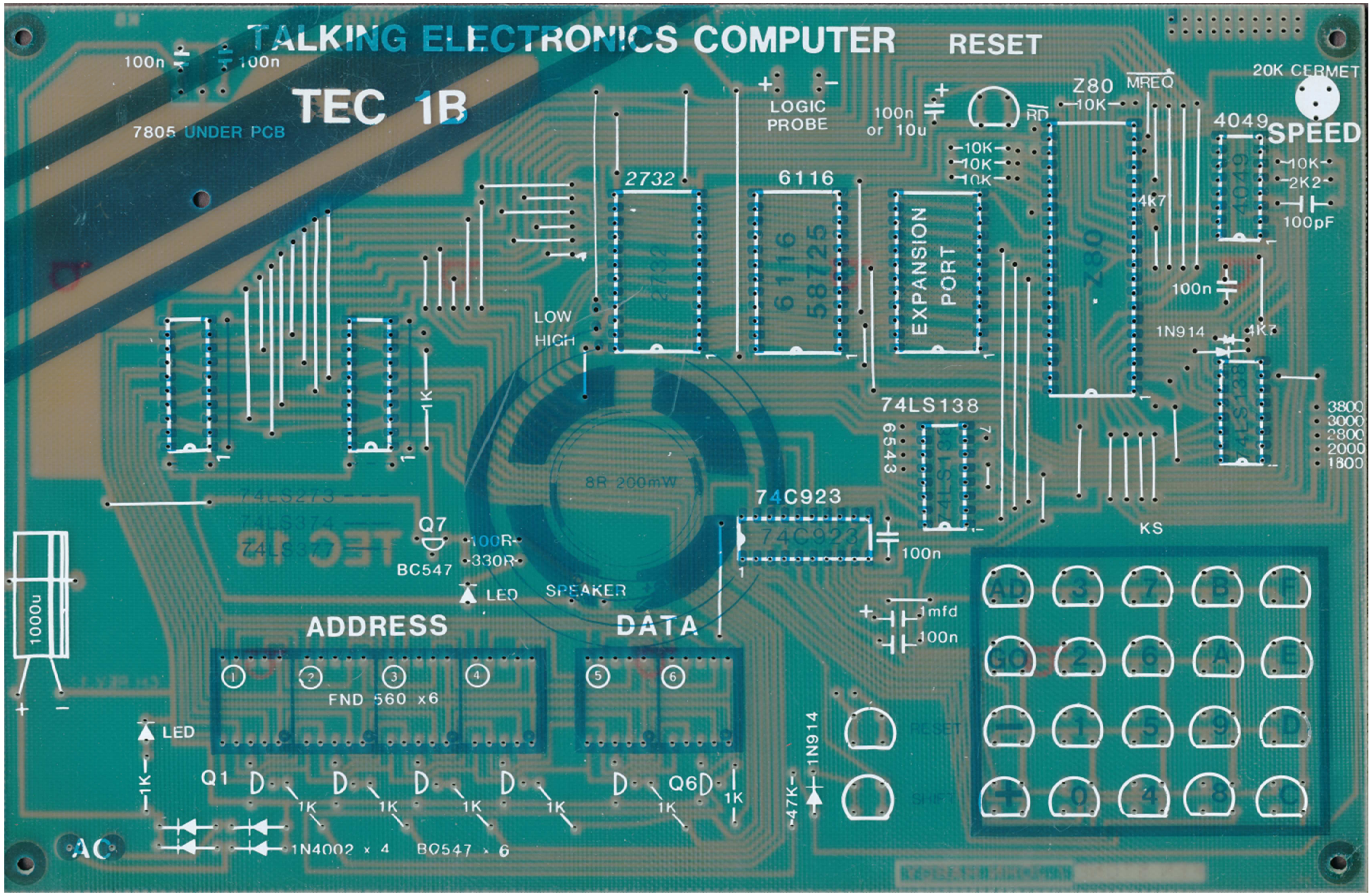
By careful programming both routines could be incorporated into one. This would require a 'check bit' and if 'set', the sub-routine would check the function keys.

Cont. P.51:

Please note we now have a reader in New Zealand interested in suppling back issues of the magazine, and maybe boards and kits. Please write to him at the following address:

Trevor Cooper,
33 York St.,
Timaru,
New Zealand.
Phone: 83787


TALKING ELECTRONICS COMPUTER
TEC 1B
MOUNT
7805
THIS SIDE
KEN STONE / JOHN HARDY
CH REV.1
KS


TALKING ELECTRONICS COMPUTER
RESET
TEC 1B
7805 UNDER PCB
LOGIC PROBE
2732
6116
EXPANSION PORT
Z80
4049
SPEED
20K CERMET
74LS138
74C923
LOW
HIGH
8R 200mW
SPEAKER
ADDRESS
DATA
FND 560 x6
Q1
Q6
Q7
BC547
1N4002 x 4
BC547 x 6
1N914
KS
1000u
AC
LED
3800
3000
2800
2000
1800