BI-PAK

ZON X-81

PROGRAMMABLE SOUND GENERATOR

INSTRUCTION MANUAL

USING THE ZonX 81 PROGRAMMABLE SOUND GENERATOR ON THE ZX 81 OR SPECTRUM

INTRODUCTION

The ZonX 81 programmable sound generator (PSG) is a complex digital device which can be easily controlled by the either computer to produce a wide range of sound effects under BASIC program control. Many effects in games need very few commands to produce the desired sounds, which results in very little additional delay in the running speed of the full program. Some effects, however, require significant program control of the PSG and it may be necessary to use the Fast mode (ZX 81 only) to reduce the delays in the unit's response to the commands.

HOW TO CONNECT THE PSG

The unit simply plugs into the ZX 81 output socket in place of the RAM pack, which may in turn also be plugged into the rear of the PSG. You can also use the printer as well as the PSG all at the same time. The PSG gets its power from the computer power supply (or the printer power supply if fitted), and does not overload it. The power supply will run slightly warmer, but this does not indicate a problem.

The PSG does not occupy memory address space and so larger RAM packs can also be used with it. The operation of add-on ports, etc., cannot be guaranteed while the PSG is connected, unless they are used with PEEK and POKE instructions in the BASIC program.

EXAMPLE PROGRAMS

These are some programs to demonstrate what can be achieved with the PSG, and to introduce you simply to the methods of controlling it. However, to show it off, the first program generates random signals to the PSG without further intervention. Type it in as shown. Line 1 in particular MUST be exactly as shown, and typed in by using keywords, rather than typing in the individual letters of "PEEK", for example. The spaces shown in line 1 are thus put in by the ZX 81 itself. (N.B. For the Spectrum see Page 22). When the whole program is typed in, RUN it. A few moments pass before sounds start, but after this anything can happen

```
 1 REM YYPEEK TO YYPEEK 3TAN
 2 GOTO 7
 3 POKE A,D
 4 POKE B,C
 5 LET X = USR E
 6 RETURN
 7 LET A = 16515
 8 LET B = 16519
 9 LET E = 16514
 10 REM RANDOM OUTPUT
 20 LET C = INT (RND * 255)
 30 LET D = INT (RND * 14)
 40 GOSUB 3
 50 FOR T = 1 TO 10
 60 NEXT T
 70 GOTO 20
```

you should notice that the PSG emits mainly noises of various sorts and only a few pure tones. This is caused by the RND numbers which tend not to favour the generation of tones, and is nothing to worry about. This is probably a good moment to adjust the volume with the

volume control to minimise the irritation to the rest of the family! When you get fed up with the noises, simply BREAK into the program. The first thing to notice when you do is that whatever the PSG was doing when you pressed BREAK, it continues to do until you make it do something else. The next program enables you to do this by loading the register number (which is called its Destination D), and its Contents C, from the keyboard.

List the original program and notice that line 1 has changed! Don't bother to change it - whatever is there is OK. In fact you'll find that the second and fourth 'Y's will be changed every time you run the PSG program. This is as it should be, and is discussed in detail later on.

Leave lines 1 to 9 as they are, and change lines 10 and onwards to the following:

```
10 PRINT AT 0,0;'KEY IN DESTINATION
REGISTER'
20 INPUT D
30 SCROLL
40 PRINT 'D= ';D
50 PRINT AT 0,0;'KEY IN CONTENTS'
60 INPUT C
70 GOSUB 3
80 PRINT AT 21,10;'C= ';C
90 GOTO 10
```

The following inputs using this short program will enable you to try some of the PSG's sounds, and show how to address the registers to generate some possibly useful sounds. Start by entering D = 8,C = 0; D = 9, C = 0; D = 10, C = 0. This will silence the PSG, ready for some systematic experiments as follows. Enter D = 0, C = 252; D = 7, C = 56 to enable tones on channels A,B,C;('enable' here means to make ready, that is, the opposite of 'disable'). Now enter D = 8, C = 15 (for full volume on channel A). You should now have a steady single tone coming from the loudspeaker. This is perhaps an appropriate moment to re-adjust the volume by hand again, because things are going to get louder! Now enter D = 2, C = 168; D = 9, C = 15 (full volume channel B), and obtain a pleasant beat of two notes. Now D = 4, C = 89; D = 10, C = 15 (full volume channel C) to obtain a rather disagreeable triple beating note between the three channels.

We will now change to envelope control of channel A while channels B and C carry on as previously set up. The word 'envelope' here refers to the way the volume of the sounds change with time, as we shall show using the following entries. Enter D = 8, C = 16 (for envelope control); channel A is immediately switched off, and B and C continue to sound together. Enter D = 12, C = 64 (to set coarse envelope period to about two seconds). Enter D = 13, C = 0; to hear channel A come on and die away.

Now bring all three channels under envelope control: D = 9, C = 16; D = 10, C = 16 (silences both B and C). Now enter D =13, C = 0 again, and all three channels come on once and die away together. Try D = 13, C = 10 for continuous rising and falling volume, or D = 13, C = 12 for rising sawtooth repetitive variation of volume, or any of the other options for register 13 shown above.

We will now switch off the tone generators and to examine the use of noise. Begin by entering

D = 6, C = 31 (for a fairly low frequency noise). Now enter D = 7, C = 7 to disable tones and enable noise in all three channels. The PSG will now be generating the last entered envelope. Enter D = 13, C = 0 for a single decay, which should resemble an explosion. Enter the same again for another one. Enter D = 12, C = 16 for a rapid decay rate, and again D = 13, C = 0 to generate a 'gunshot'.

Enter D =12,C = 0;D =11,C = 128;D = 13,C = 10. You should now have a helicopter sound. It is interesting at this stage to try all the valid loads to D = 13, to find out what is meant by the entries in the table later on.

We will now try the effect of mixing noise and tones in one channel to start with and then add other channels one by one. Load D = 9, C = 0; D = 10, C = 0 to set the volume of channels B and C to zero, envelope control off. Enter D = 7, C = 0 to enable noise and tones in all three channels. Now D = 8, C = 15 will generate a mixture of tones and noise in channel A only.

Having shown how to control the PSG by hand, the next section describes in much more detail what is actually going on inside the device, and will lead up to a description of how to set it up to generate any sound you think up, and which is within its ability to produce.

The final section of the leaflet lists a number of sample programs to generate such sounds as explosions, bombs, laser guns, and whistles, which you can add to your existing game programs if you wish. Two complete programs to use it as a simple electronic organ, and as a bell-ringer, are also given.

HOW THE PSG WORKS.

The PSG contains three separate sound channels which are separately controllable in various modes for frequency and volume. The channels are labelled A,B,C and their operation is controlled by separate registers within the device. The allocation of these control register numbers, and their function, is described below.

Each channel may be used to generate a single tone whose frequency must be set up by the user by loading a pair of registers: 0 and 1 for channel A, 2 and 3 for channel B, 4 and 5 for channel C. Alternatively each channel may be used to transmit noise, whose frequency (or 'note') is the same for all three channels, and is set up in a separate register (6).

A single register (7) is used to control the types of sound which each of the three channels generates, i.e. tones, noise, or both. Any combination can be selected by sending the correct code to register 7, as will be described later.

The combination selected is combined in the PSG and sent to the loudspeaker via a manual volume control which sets the overall volume of all the sounds produced. The volume can also be controlled by program instructions using registers 8, 9, and 10. These registers control channels A, B, and C respectively. Two options are available in each: either programmed envelope control or programmed fixed volume. In fixed volume mode , sixteen levels are available between zero (silence) and 15 (full). In envelope control mode, the volume always varies between maximum and minimum level. The rate at

which the volume is changed is set up in registers 11 and 12, and will be described in the section entitled 'Internal operation of the PSG'. Finally, register 13 controls the way in which the envelopes operate. Effects such as single decays (to simulate gunshots, bells, etc.), repetitive sweeps (helicopters, sonars) and single build-ups (approaching aircraft, etc.), can all be set up.

The setting for type of envelope applies to all those channels that have been enabled (i.e. set up) for envelope control. Thus if two channels are enabled for a single build-up in volume by a load to register 13, both will both increase together.

The contents loaded in any register remain until they are changed by a new load. It is therefore only necessary to change the contents of registers where a change is actually needed. Effects such as explosions, for example, can be achieved by a single load to register 13 to start a new envelope.

CONTROL BY THE ZX 81

Before describing the functions of the registers in detail, and how to use them, the method of controlling the PSG is described. The register number and its contents are defined in the ZX 81 BASIC program, and these two data are passed to the PSG by a very simple machine code routine. This is stored in a REM statement which must form the first line of any program in which you want to use the sound generator.

The following lines are suggested as the first in any program using the PSG, because they result in very fast access to the PSG, and easy use of the single subroutine. Line 1 is shown exactly as it first appears on the screen, having typed it in using the appropriate key-words. No spaces should be entered, because the ZX 81 inserts them where required.

```
1 REM YYPEEK TO YYPEEK 3TAN
2 GOTO 7
3 POKE A,D
4 POKE B,C
5 LET X = USR E
6 RETURN
7 LET A = 16515
8 LET B = 16519
9 LET E = 16514
10 REM YOUR PROGRAM STARTS HERE
```

The register number is the Destination of the data, and is stored in D, by a LET D = data statement in the main program which follows. Likewise, the Contents desired are passed by loading up C in a statement such as LET C = data.

The PSG loader machine code routine is most effectively called by a BASIC subroutine, because it will be needed frequently in a complicated sequence of sounds, and also because the subroutine can be faster to execute if it is placed near the beginning of the BASIC program. The number 16514 is the address in the ZX 81's memory of the machine code routine stored in the REM statement in the first line of the program. The numbers 16515 and 16519 actually point to the second and fourth "Y"s in the REM statement at line 1, and every time the subroutine is executed the contents of locations 16515 and 16519 are changed. If the program is

listed after running, the REM statement will be found to have altered, and the second and fourth 'Y's replaced with some other character or command. There is nothing wrong in this - it does not matter what is stored in these locations during listing, and whatever is there can be stored on tape. There is no need to alter it back to its original state.

INTERNAL OPERATION OF THE PSG.

It is necessary to understand a little of the internal operation of the PSG to take full advantage of its facilities, which were summarised above. The exact function of each of the internal registers, and the numbers which must be loaded to achieve the desired effects, will be described.

REGISTERS D0 and D1

These registers tune channel A of the PSG. Register 1 is coarse tuning, and register 0 is fine tuning. The combined contents set the period of the note (that is, the length of a cycle), so that large numbers set up lower notes, and small numbers set up higher notes. The actual note can be calculated using the following equation:

$$f = 1625000/(16*(256*C1 + C0))$$

C1 and C0 in the equation are the actual contents of registers D1 and D0. Register D0 may contain integer (i.e. without a fractional part) values in the range 0 to 255, and register D1 may contain integer values in the range 0 to 15. If both are set to zero no sound is generated, but it is better to switch the sound off using register 8. The highest possible note obtainable with C0 = 1, and C1 = 0, is about 101 kHz - totally inaudible! Middle C is usually taken to be 261.624 Hz, so the required values of $256*C0 + C_1$ are 388.21, rounded to 388. Now C1 is 256 times C0, and so the contents of the two registers are calculated as C1 = 1*(256) and C0 = (388 - 256) = 132. So D1 is loaded with 1, and D0 with 132. Channel A will then produce 261.759 Hz if enabled (i.e. turned on in register 7).

REGISTERS D2 AND D3

These operate on channel B in exactly the same way as registers D0 and D1 operate on channel A.

REGISTERS D4 AND D5

These operate exactly as above, but on channel C.

REGISTER D6

This register controls the period of the noise generator on all three channels, if enabled, that is, set up in register 7. The contents can range from 1 to 31. Larger numbers give lower pitched noise ('explosion-like'), and smaller numbers give higher pitched noise ('escaping steam').

REGISTER D7

This register is used to enable and disable noise and tones.

Channel A noise is DISABLED by adding 8 to D7.

Channel B noise is DISABLED by adding 16 to D7.

Channel C noise is DISABLED by adding 32 to D7.

Channel A tone is DISABLED by adding 1 to D7.

Channel B tone is DISABLED by adding 2 to D7.

Channel C tone is DISABLED by adding 4 to D7.

Thus loading D7 with 0 enables noise and tones from all three channels. Similarly, loading 7 (ie 1+2+4) enables noise only from all three channels, and loading 56 (ie 8+16+32) enables tones only from all three channels.

REGISTERS D8, D9, D10

These three registers set up the volume of channels A,B,C, respectively. Loading numbers in the range 0 to 15 set up the volume between off (0) and maximum (15). Loading 16 sets up the selected channel for envelope control by register 13 (see below).

REGISTERS D11 AND D12

These two registers control the rate of variation of volume in channels enabled for envelope control in registers 8,9,10. The period of the variation is given by the equation:

$$p = 162500/(256*(256*C12 + C11))$$

The contents (C11 and C12) of D11 and D12 can both have values from 0 to 255. If both are loaled with 255, a very slow change of volume is achieved, in which the variation takes about 10 seconds to run its full range.

REGISTER D13

This register controls the type of envelope to be produced. Its operation is best described by listing the effects of all the different loadings to D13, as follows:

CONTENTS	EFFECT
0	Single decay (gunshot,etc.)
6	Single rise, then quiet.
8	Falling sawtooth continuous
10	Continuous rise and fall, starts with falling volume.
11	Single falling to zero then back to maximum.
12	Rising sawtooth continuous.
13	Rising to full volume, then steady.
14	As 10, but starting with rising volume.

The use of 0 is probably the most useful because it generates a single decaying sound (whether noise or tone).

USEFUL PROGRAMS

Here are some useful sequences of programs which make sounds you may wish to try out, or to use in your own programs. It is assumed that the previous lines 1 to 9 are present at the beginning of any program using these examples. Also notice that the method of linking them together is up to you when you use them. Don't forget to include a RETURN if you call them by a GOSUB, or add a GOTO to get you back to the main program.

CLEAR ALL REGISTERS

This segment of program shuts off all sounds by clearing all registers.

```
100 LET C = 0
110 FOR D = 0 TO 13
120 GOSUB 3
130 NEXT D
```

Note here that you don't need to have the LET C = 0 inside the loop: once set, the same value will be used each time the GOSUB 3 is called, for all the consecutive values of D. A faster version of this, which turns off all three channels is to make line 110 FOR D = 8 TO 10. It does leave all the other channels in an unpredictable state, however.

WHISTLING BOMB

This program sweeps the frequency of the tone to sound like a falling bomb. Let's hope this is the only sort we ever get to hear! The comments in brackets are to help in understanding what's going on, so don't try to include them in the program. It sounds better run in FAST mode.

```
200 LET D = 7
210 LET C = 62 (A only enabled)
220 GOSUB 3
230 LET D = 8
240 LET C = 15 (A full volume)
250 GOSUB 3
260 LET D = 0 (A frequency)
270 FOR C = 48 TO 192
280 GOSUB 3
290 NEXT C (to sweep frequency)
300 LET D = 7
```

```
310 LET C = 63
320 GOSUB 3 (to silence A)
```

EXPLOSION

This effect sounds good immediately following the falling bomb! FAST mode sounds better.

```
400 LET D = 6
410 LET C = 31
420 GOSUB 3
430 LET D = 7
440 LET C = 7 (enable noise only)
450 GOSUB 3
460 LET C = 16 (for envelope control)
470 FOR D = 8 TO 10 ( i.e. all three)
480 GOSUB 3
490 NEXT D
500 LET D = 12 (envelope period)
510 LET C = 56 (a slow decay)
520 GOSUB 3
530 LET D = 13
540 LET C = 0 (a single decay)
550 GOSUB 3
```

GUNSHOT

All that is needed is to change the value of C in line 410 to 1, and the value of C in line 510 to 16 in the explosion program given above.

'LASER GUN'

This program sweeps the noise frequency while the volume is dying away to produce a descending 'laser' sound. It sounds better in FAST mode.

```
600 LET C = 0
610 FOR D = 8 TO 10
620 GOSUB 3
630 NEXT D
640 LET D = 7
650 LET C = 55 (noise only channel A)
660 GOSUB 3
670 LET D = 8
680 LET C = 16 (envelope control)
690 GOSUB 3
700 LET D = 12
710 LET C = 15
720 GOSUB 3
730 LET D = 13
740 LET C = 0 (a single decay)
750 GOSUB 3
760 LET D = 6
770 FOR C = 1 TO 31 STEP 3
780 GOSUB 3
790 NEXT C
800 STOP
```

WOLF WHISTLE

This only sounds right in FAST mode!

```
600 LET D = 6
610 LET C = 1 (for a hissing sound)
620 GOSUB 3
630 LET D = 7
640 LET C = 45 (A tones, B noise)
650 GOSUB 3
660 LET D = 8
670 LET C = 15 (full volume on A)
680 GOSUB 3
690 LET D = 9
700 LET C = 6 (for quiet 'hissing')
710 GOSUB 3
720 LET D = 0
```

```
730 FOR C = 100 TO 30 STEP -3
740 GOSUB 3
750 NEXT C
760 LET D = 7
770 LET C = 63 (silence!)
780 GOSUB 3
790 FOR T = 1 TO 5(for a pause)
800 NEXT T
810 LET C = 46
820 GOSUB 3 (D is7: A + B on again)
830 LET D = 0
840 FOR C = 60 TO 30 STEP -3
850 GOSUB 3
860 NEXT C
870 FOR C = 30 TO 100 STEP 3
880 GOSUB 3 (D is still 0!)
890 NEXT C (the final falling note)
900 LET C = 0
910 FOR D = 8 TO 10
920 GOSUB 3 (turns off all sounds)
930 NEXT D
940 STOP
```

BELL-RINGING

This program rings what is known as 'Plain Bob Maximus' on twelve bells. The value of Z may be changed to alter the number of bells, but don't use odd numbers. As before, lines 1 to 9 are assumed to be present so that the GOSUBs make sense and the PSG is correctly driven. The program takes several minutes to repeat, because the sequence changes after every twelve changes. This also sounds better in FAST mode, although it is not difficult to adjust the times to compensate for running in SLOW.

```
10 LET Z = 12 (the number of bells)
20 LET L = 1
30 LET R = 600 (ringing subroutine)
40 LET V = 800 (changing subroutine)
50 LET D = 7
60 LET C = 62 (channel A tones only)
70 GOSUB 3
80 LET D = 8
90 LET C = 16 (envelope control)
100 GOSUB 3
110 LET D = 9
120 LET C = 0
130 GOSUB 3
140 LET D = 10
150 GOSUB 3 (C is still 0)
160 LET D = 12
170 LET C = 15
180 GOSUB 3
200 DIM B(12) (the number of bells)
210 LET B(1) = 84
220 LET B(2) = 89
230 LET B(3) = 100
240 LET B(4) = 112
250 LET B(5) = 126
260 LET B(6) = 134
270 LET B(7) = 150
280 LET B(8) = 168
290 LET B(9) = 189
300 LET B(10) = 200
310 LET B(11) = 225
320 LET B(12) = 252
330 REM THE RINGING PROGRAM:
400 FOR Y = 1 TO Z - 1
410 FOR N = 1 TO 2
420 GOSUB R (to ring them)
430 GOSUB V (to do the changes)
440 NEXT N
450 FOR T = 1 TO 2
460 NEXT T (a short delay)
470 NEXT Y
```

```
480 LET N = 3 (for last change only)
490 GOSUB R (ring twelfth change)
500 GOSUB V (all except 1st two)
510 GOTO 400 (to continue)
the ringing subroutine:
600 FOR J = L TO Z
610 LET D = 0
620 LET C = B(J)
630 GOSUB 3 (to define the note)
640 LET D = 13
650 LET C = 0
660 GOSUB 3 (to make it sound)
670 FOR T = 1TO 10
680 NEXT T (for a delay)
690 NEXT J
700 RETURN
the changing subroutine:
800 FOR S = N TO Z - L
810 LET K = B(S)
820 LET B(S) = B(S+L)
830 LET S = S+L
840 LET B(S) = K
850 NEXT S (to swap next pair)
860 RETURN
```

ELECTRONIC ORGAN

This program makes the keys 1 to 0 into a simple keyboard. The sound persists while you keep your finger on the key by continuously loading 0 to register 13. When you lift your finger off, the sound is allowed to decay, as if it was an organ. Notice that the notes in lines 240 to 330 are the same as those used in the bell-ringer program. Run it in FAST mode.

```
100 LET C = 0
110 FOR D = 1 TO 11
120 GOSUB 3 (clear registers)
```

```
130 NEXT D
140 LET D = 7
150 LET C = 62 (tone only ch. A)
160 GOSUB 3
170 LET D = 8
180 LET C = 16 (ch. A envelope)
190 GOSUB 3
200 LET D = 12
210 LET C = 30 (envelope period)
220 GOSUB 3
230 DIM B(10)
240 LET B(1) = 100
250 LET B(2) = 252
260 LET B(3) = 225
270 LET B(4) = 200
280 LET B(5) = 189
290 LET B(6) = 168
300 LET B(7) = 150
310 LET B(8) = 134
320 LET B(9) = 126
330 LET B(10) = 112
340 LET G = (CODE INKEY$)-27
350 IF G<1 OR G>10 THEN GOTO 340
360 LET D = 0 (when a key pressed)
370 LET C = B(G) (select the note)
380 GOSUB 3
390 LET D = 13
400 LET C = 0
410 GOSUB 3 (play the note)
420 GOTO 340 (keep looking for keys)
```

WHAT THE MACHINE CODE DOES

The machine code in the REM statement in the first line actually uses the data passed by the two POKEs in lines 3 and 4 as follows. The PSG "looks" to the ZX 81 as if it consists of two ports. The first, with the port address DF (hexadecimal, normally written DFH), is the destination register defined by the contents of the microprocessor's accumulator. The second, with the port address 0FH, is the contents of the previously chosen destination register.

The assembly language version of the REM statement after loading register 7 with 63 (3FH) is as follows:

```
16514  3E LD A, 07
16515  07
16516  D3 OUT (0DFH),A
16517  DF
16518  3E LD A,3FH
16519  3F
16520  D3 OUT (0FH),A
16521  0F
16522  C9 RET
```

You can verify that the hexadecimal codes shown translate into the words in the REM statement from the list given in the back of the ZX 81 manual. The use of YY in the REM statement is chosen merely because the second and fourth "Y"s are needed to make space for the real data to be POKEd to addresses 16515 and 16519 later on.

ADDING ZonX 81 MACHINE CODE TO OTHER PROGRAMS

If the ZonX 81 is to be used with other machine code routines also stored in the first line REM statement, it is a simple matter to relocate the addresses of the ZonX 81 code so that both routines are available to you without interference. Since most machine code is not relocatable, the best way to do this is to add the nine characters of the ZonX 81 code at the end of any other machine code entry in the first line REM statement.

The simplest way to enter the ZonX 81 machine code is to do it after the other machine code is already in place in the program. It must, of course, be a REM statement in the first line, but this is the only condition which must be fulfilled. All you have to do is put the cursor onto line 1, and EDIT it. Move the cursor to the end (this is the worst part if the REM statement is very long!), and then simply enter the characters YYPEEK TO YYPEEK 3TAN and follow them with a NEWLINE. Remember they must be typed in using keywords rather than typing in "PEEK", for example. The extra code is now part of the program, and can be saved along with the rest of the program. Lines 2, 3, 4, 5, and 6 should be entered as shown on page 2.

It is necessary to change lines 7, 8, and 9 to compute the new values for A, B, and E, which were simply assigned in lines 7, 8, and 9 of the program given on page 2. This is most reliably done by the machine itself, using the pointers and line lengths described in Chap. 27 (particularly page 171) of the ZX 81 manual. To do these computations, use the following lines 7, 8, and 9 instead of the ones given in the ZonX 81 manual.

```
   LET A = 16504 + (256 * PEEK 16512 + PEEK
 16511)
 8 LET B = A + 4
 9 LET E = A - 1
```

The number 16504 is arrived at by subtracting 9 (the length of the ZonX 81 machine code) from the address of the first character (REM) in the first line. The two addresses 16512 and 16511 contain the length of the first line, including the ZonX 81 code and a NEWLINE. When this length is added to 16504, the address formed is that of the second "Y" in the ZonX 81 machine code. The other two addresses are then simply calculated relative to the address calculated for A. Note that the new lines 7, 8, and 9 work correctly for any first line REM statement of any length, including the simple one given on page 2 of this manual.

USING THE PSG ON THE SINCLAIR SPECTRUM

The Spectrum computer has a simple way of passing data to external devices such as the ZonX-81, using the OUT A,B form of instruction. This means that machine code is not required on the Spectrum. The BASIC subroutine which passes the Destination register number and Contents is as follows:

```
2 GOTO 7
3 OUT A,D
4 OUT B,C
6 RETURN
7 LET A = 65535
8 LET B = A - 128
10 REM ALL OTHER PROGRAMS AS FOR ZX-81
```

Note that the subroutine is still at line 3, and will be called correctly by all the programs given in this manual. Lines 1, 5, and 9 in the ZX-81 version are no longer needed.

printed and published by BI-PAK, Ware, Herts.