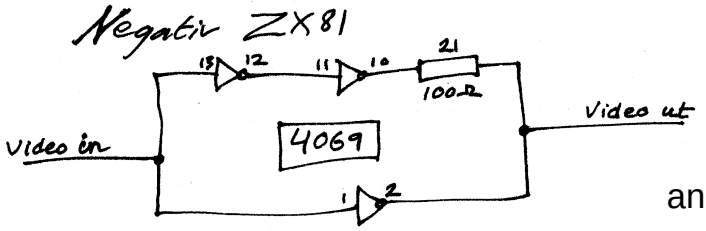
Collected Articles and Tips about Sinclair ZX81

by Mikael O. Bonnier

2nd Ed



ur EV nr 8 sid 71

Lund, Sweden 1986-

but Bjärred -1985

and Ängelholm (F10) 1985-1986

INSTALLATION NOTES FOR THE 56K SPECIAL RAMPACK FOR THE ZX81

1) Please power off the computer.

2) Clip the blue connector on to the expansion port of the ZX81.

3) Push firmly to ensure that the connector is fully inserted and the ridge on the front case cover clamps securely to the sinclair case.

4) Power on the computer. The screen will go blank for around 3 seconds, and the cursor will then reappear. You will notice that the machine with the new Ram is slower than the 1K machine. This is due to the fully expanded screen; most operations have to update the screen contents, and the operating system has to re-write the whole screen every time - the speed is therefore slower.

5) Load from tape your favourite programs and check that everything is working perfectly.

6) Please note that most existing software will run without requiring any "poke" priory to the load operation.

EXTRA MEMORY MEANS EXTRA PERFORMANCE:

Usually the Sinclair computer sets its RAMTOP at 32768 decimal or 8ØØØ H. The SPECIAL RAMPACK offers the machine an absolute RAMTOP at 65535 decimal or FFFF hex. Therefore, you can set the working RAMTOP at any level you like although the following rules should be observed:

1) The RAMTOP is equal to PEEK 16388 + 256 * PEEK 16389.

2) The BASIC RAM starts at 16384 decimal or 4ØØØ H.

3) The amount of RAM reserved by BASIC is therefore RAMTOP - START OF BASIC RAM.

   This can be calculated by:

   PRINT PEEK 16388 + 256 * (PEEK 16389 - 64).

4) To decide what RAMTOP must be used you must first estimate the extent of:

   a) The program text.
   b) The variable area.

   Add the 2 things together and allow some spare headroom and set the RAMTOP.

5) The setting of RAMTOP does not change the time taken to load or save but simply allows you to use the part of memory unused by basic for other purposes such as swapping programs, storing screen-fulls of printing information, graphic representation, character sets, number arrays, extra character arrays, machine code routines etc . .

6) Example of RAMTOP:

16K basic: No change (PEEK 16389 = 128), extra memory = 32K + 8K = 40K bytes.

32K basic = POKE 16389, 192 New Line NEW N/L

extra memory = 16K + 8K = 24K

48K basic = POKE 16389,255 N/L NEW N/L

extra memory = 8K bytes.

So by setting a new RAMTOP, you can give the ZX81 more space, power and bigger and better basic programs.

The maximum character array for a special Ram Pack is around

The maximum number array is 93ØØ elements.

RUNNING MACHINE CODE ON THE SPECIAL RAMPACK

```
  10 LET N=32+PEEK 16406+256*PEE
K 16407
  20 LET A$="CD 10 20 28 04 CD 1
0 20 EB ED B0 C9 00 00 00 00 05
01 11 D6 02 21 00 80 19 10 FD EB
 44 4D 2A 0C 40 C9"
  30 LET S=8192
  40 FOR I=1 TO LEN A$-1 STEP 3
  50 POKE S+I/3,((PEEK (N+I)-28)
*16+(PEEK (N+I+1)-28))
  60 NEXT I
  70 LET I=S+5
  80 LET PAGE=S+17
 100 PRINT "THE SCREEN PAGE REQU
IRES 726 BYTES TO STORE-STARTING
 FROM LOCATION 32679"
 105 PRINT
 106 PRINT "DEPENDING ON THE AMO
UNT OF MEMORY YOU RESERVE,A NUMB
ER OF PAGES CAN BE FREELY WRITTE
N INTO WITHOUT INTERFERENCE TO Y
OUR PROGRAM"
 110 PRINT AT 10,10;"__________"
 120 PRINT "ONLY 22 PAGES ARE AV
AILABLE ON A 32K RAMPACK"
 125 PRINT "UP TO 44 PAGES CAN B
E STORED WIH A 48K OR +"
 130 PRINT
 135 PRINT "WRONG ANSWER WILL CA
USE PROGRAM TO CRASH"
 137 PRINT AT 20,1;"PLEASE ANSWE
R WITH 32-48-56-OR-64"
 140 INPUT N
 150 LET TOP=(22 AND (N=32))+(44
 AND (N=48))+(52 AND (N=56))+(66
 AND (N=64))
 160 CLS
 170 PRINT "YOU SAY ";N;"K BYTES
 SO ";TOP;" PAGES ARE RESERVED.P
LEASE ENTER DETAILS "
 175 PRINT "PRESS >= TO GO TO NE
XT PAGE"
 176 PRINT "PRESS <= TO VIEW LAS
T PAGE"
 177 PRINT "PRESS  LPRINT TO COP
Y"
 180 PRINT AT 20,5;"PRESS A KEY
WHEN READY"
 190 IF INKEY$="" THEN GOTO 190
 200 IF TOP>44 THEN GOTO 500
 210 GOSUB 230
 220 GOTO 200
 230 FOR N=1 TO TOP
 240 POKE PAGE,N
 250 CLS
 255 PRINT "PAGE=";N;" BLOCK STA
RTING AT ";PEEK (S+23)
 260 IF INKEY$="" THEN GOTO 260
 270 IF INKEY$<>">=" THEN GOTO 3
00
 280 RAND USR S
 286 GOTO 450
 300 IF INKEY$<>"<=" THEN GOTO 3
50
 310 LET N=N-1
 320 IF N=0 THEN RETURN
 330 POKE PAGE,N
 340 RAND USR I
 345 GOTO 410
 350 IF INKEY$<>" LPRINT " THEN
GOTO 400
 360 COPY
 370 GOTO 410
 400 PRINT INKEY$;
 410 IF INKEY$<>"" THEN GOTO 410
 420 GOTO 260
 450 NEXT N
 460 RETURN
 500 LET EXTRA=TOP-44
 510 LET TOP=44
 520 GOSUB 230
 530 IF EXTRA>8 THEN GOTO 600
 540 POKE S+23,33
 550 LET TOP=EXTRA
 560 GOSUB 230
 570 POKE S+23,64
 580 GOTO 510
 600 REM USING SECODARY MEMORY
 610 POKE 14339,130
 620 POKE 14338,255
 630 LET TOP=44
 640 GOSUB 230
 650 POKE 14338,127
 660 REM SWITCH ON SEC.MEM.
 670 LET TOP=22
 680 GOSUB 230
```

RUNNING MACHINE CODE ON THE SPECIAL RAMPACK

Running machine code on the Special RAMPACK is a pleasure because you can enjoy 8K of absolute RAM, untouched by basic load and save. This memory is present between 8192 decimal and 16383 decimal. The program listed on the last page allows you to enter machine code program at any location.

APPLICATION EXAMPLES:

We list below some of the utilities for the user who wants to write machine code from basic without any extra tool (e.g. machine code monitor):

BINARY MERGE:

This routine allows you to copy any machine code from memory to a basic string or a basic REM statement. Therefore it can be saved at the same time with your program. This allows you to relocate it back when you want it.

```
1 REM ........
2 REM PLEASE LEAVE 60 SPACES FOR MACHINE CODE IN LINE1
1Ø DIM A$ (100)
15 GOSUB 200
20 PRINT "PLEASE NOTE DOWN"
30 PRINT AT 2,3; "USR 16514: MOVES CONTENTS OF A$ TO TARGET WHOSE
   ADDRESS IS AT LOCATION 16569 (HI) AND 16568 (LOW)"
40 PRINT AT 6,3; "USR 16519: PERFORMS THE OPPOSITE FUNCTION"
50 PRINT AT 9,3; "USR 16576 MOVES THE REM CONTENTS FOLLOWING THE USR
   TO THE TARGET WHOSE ADDRESS IS AT 16567 (HI) AND 16566 (LOW)"
60 PRINT AT 13,3; "USR 16581 PERFORMS THE OPPOSITE FUNCTION"
70 PRINT AT 15,3; "THESE UTILITIES ARE PROVIDED TO FACILITATE
   LOAD/SAVE BINARY FILES ON TAPE, BY SETTING THE TARGET ADDRESSES
   YOU CAN MERGE SEVERAL BINARY FILES THAT CAN BE USED DIRECTLY TO
   PROGRAM EPROMS (SUPER Z DMOTAPE)"
80 RAND USR 16514
90 REM WRITE TARGET MEMORY WITH 100 SPACES FOR A NICE PRESENTATION
11Ø RAND USR 16576
12Ø REM PLEASE ENTER YOUR STATEMENT HERE, LINE 11Ø MOVES THE CONTENTS
    OF LINE 12Ø TO THE TARGET MEMORY
13Ø REM WE NOW WRITE A$ WITH THE PREVIOUS DATA IN LINE 14Ø
14Ø RAND USR 16519
```

```
15Ø PRINT A$
16Ø STOP
2ØØ LET N=32 + PEEK 16406 + 256 *PEEK 16407
21Ø LET B$="CD.90.40.2E.04.CD.90.40.EB.ED.B0.C9.00.00.2A.10.
    40.23.23.23.23.4E.23.46.23.11.00.80.C9.00.2A.16.40.23.23.
    23.23.23.E5.3E.76.01.00.00.03.23.BE.20.FB.E1.0B.11.00.80.C9"
22Ø PRINT AT 10,5; "MACHINE CODE LOADING"
23Ø FOR I=1 TO LEN B$-1 STEP 3
24Ø POKE S+1/3,((PEEK (N+1)-28)*16+(PEEK (N+I+1)-28))
25Ø NEXT I
26Ø CLS
27Ø RETURN
```

READ/WRITE 16K PROGRAM:

```
1Ø  LET N=32+PEEK 16406+256*PEEK 16407
2Ø  LET A$="CD.E7.02.CD.10.20.1.1.09.80.ED.B0.C3.07.02.00.00.06.
    01.11.00.40.21.09.00.19.10.FD.E5.11.0B.00.19.5E.23.56.21.09.
    C0.1.9.44.4D.E1.C9"
3Ø  LET S=8192
4Ø  FOR I=1 TO LEN A$-1 STEP 3
5Ø  POKE S+I/3, ((PEEK (N+I)-28)*16+(PEEK (N+I+1)-28))
6Ø  NEXT I
7Ø  LET TARGET=8200
8Ø  LET SELECT=8209
9Ø  PRINT "TARGET=";PEEK TARGET;" (TARGET ADDRESS*256 BYTES IS
    WHERE YOU SAVE YOUR PROGRAM"
1ØØ PRINT "NORMAL RUNNING PROGRAM, TARGET=64"
11Ø PRINT "PROGRAM 2, TARGET=128"
12Ø PRINT "PROGRAM 3, TARGET=192, ONLY FOR 48K/56K/64K"
13Ø PRINT "PROGRAM 4, TARGET=128, SECONDARY MEMORY SET, ONLY FOR SUPER Z"
14Ø PRINT "SELECT=";PEEK SELECT;"(THIS IS THE CHOSEN PROGRAM, EITHER THE
    RUNNING PROGRAM(1) OR PROGRAM 2 OR 3"
15Ø PRINT "PRESS A KEY TO COPY THIS PROGRAM IN BLOCK 2 AND 3"
16Ø IF INKEY$="" THEN GOTO 16Ø
17Ø POKE TARGET, 128
18Ø RND USR S
19Ø POKE TARGET, 192
2ØØ RND USR S
21Ø CLS
22Ø PRINT AT 10,10;"ALL DONE"
```

23Ø PRINT AT 15,3;"YOU CAN ERASE THIS PROGRAM BY NEW OR BY LOADING A NEW PROGRAM"

24Ø PRINT "TO CALL A PROGRAM FROM MEMORY PLEASE NOTE ON A PIECE OF PAPER POKE 8200,64(TARGET LOADED WITH THE ADDRESS OF RUNNING PROGRAM), FOLLOWED BY POKE 8209,2(OR 3). NOTE THAT SELECTING A NON EXISTING PROGRAM WILL CAUSE THE SYSTEM TO CRASH. FINALLY TYPE RND USR 8192."

SCREEN LOGGER:

This allows you to use the extra memory as buffer for displays. If you have a printer, you can print in the same loop as many pages as you have set aside the memory for

```
  10 LET N=32+PEEK 16406+256*PEE
K 16407
  20 LET A$="CD 10 20 28 04 CD 1
0 20 EB ED B0 C9 00 00 00 00 06
01 11 D6 02 21 00 80 19 10 FD EB
 44 4D 2A 0C 40 C9"
  30 LET S=8192
  40 FOR I=1 TO LEN A$-1 STEP 3
  50 POKE S+I/3,((PEEK (N+I)-28)
*16+(PEEK (N+I+1)-28))
  60 NEXT I
  70 LET I=S+5
  80 LET PAGE=S+17
 100 PRINT "THE SCREEN PAGE REQU
IRES 726 BYTES TO STORE-STARTING
 FROM LOCATION 32678"
 105 PRINT
 106 PRINT "DEPENDING ON THE AMO
UNT OF MEMORY YOU RESERVE,A NUMB
ER OF PAGES CAN BE FREELY WRITTE
N INTO WITHOUT INTERFERENCE TO Y
OUR PROGRAM"
 110 PRINT AT 10,10;"■■■■■■■■■■■■■■■■"
 120 PRINT "ONLY 22 PAGES ARE AV
AILABLE ON A 32K RAMPACK"
 125 PRINT "UP TO 44 PAGES CAN S
E STORED WIH A 48K OR +"
 130 PRINT
 135 PRINT "WRONG ANSWER WILL CA
USE PROGRAM TO CRASH"
 137 PRINT AT 20,1;"PLEASE ANSWE
R WITH 32-48-56-OR-64"
 140 INPUT N
 150 LET TOP=(22 AND (N=32))+(44
 AND (N=48))+(56 AND (N=56))+(66
 AND (N=64))
 160 CLS
 170 PRINT "YOU SAY ";N;"K BYTES
 SO ";TOP;" PAGES ARE RESERVED.P
LEASE ENTER DETAILS "
 175 PRINT "PRESS >= TO GO TO NE
XT PAGE"
 176 PRINT "PRESS <= TO VIEW LAS
T PAGE"
 177 PRINT "PRESS  LPRINT TO COP
Y"
 180 PRINT AT 20,5;"PRESS A KEY
WHEN READY"
 190 IF INKEY$="" THEN GOTO 190
 200 IF TOP>44 THEN GOTO 500
 210 GOSUB 230
 220 GOTO 200
 230 FOR N=1 TO TOP
 240 POKE PAGE,N
 250 CLS
 255 PRINT "PAGE=";N;" BLOCK STA
RTING AT ";PEEK (S+23)
 260 IF INKEY$="" THEN GOTO 260
 270 IF INKEY$<>">=" THEN GOTO 3
00
 280 RAND USR S
 286 GOTO 450
 300 IF INKEY$<>"<=" THEN GOTO 3
50
 310 LET N=N-1
 320 IF N=0 THEN RETURN
 330 POKE PAGE,N
 340 RAND USR I
 345 GOTO 410
 350 IF INKEY$<>" LPRINT " THEN
GOTO 400
 360 COPY
 370 GOTO 410
 400 PRINT INKEY$;
 410 IF INKEY$<>"" THEN GOTO 410
 420 GOTO 260
 450 NEXT N
 460 RETURN
 500 LET EXTRA=TOP-44
 510 LET TOP=44
 520 GOSUB 230
 530 IF EXTRA>8 THEN GOTO 600
 540 POKE S+23,33
 550 LET TOP=EXTRA
 560 GOSUB 230
 570 POKE S+23,64
 580 GOTO 510
 600 REM USING SECODARY MEMORY
 610 POKE 14339,130
 620 POKE 14338,255
 630 LET TOP=44
 640 GOSUB 230
 650 POKE 14338,127
 660 REM SWITCH ON SEC.MEM.
 670 LET TOP=22
 680 GOSUB 230
```

Please note that these listings show lots of printing lines. These can be omitted once you get used to the new facilities and if you wish to incoporate them to your existing programs, usually only the first 1Ø lines have to be kept.

WILLIAM STUART SYSTEMS

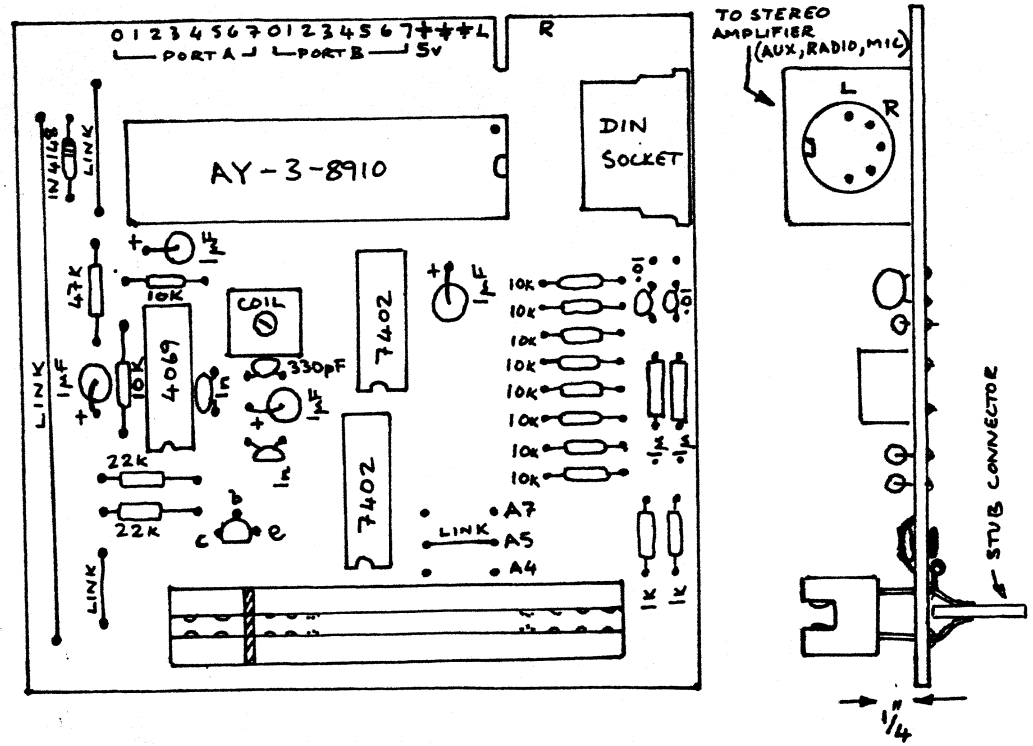
INPUT/OUTPUT PORT and MUSIC SYNTHESISER

for

ZX81 and SPECTRUM

Please note: This board connects, via DIN cable, to external amplifier or tape recorder. Certain tape recorders work as ampliers while recording. Always check your amplifier works by touching DIN cable pins- you should hear a loud hum if it is operating.

1. Component Layout



2. Parts List (Mk III boards)

Qty	Part
1	Printed Circuit Board
1	Stub Connector Board
1	23 way edge connector
1	DIN Socket (audio out)
1	40 pin IC socket
1	AY-3-8910 Sound IC
2	74LS02
1	4069
1	Coil L132
1	NPN Transistor
1	Diode 1N4148
4	1uF electrolytic
2	0.1uF
2	0.01uF
2	0.001uF (=1nF)
1	330pF
1	47k (yellow mauve orange)
2	22k (red red orange)
10	10k (brown black orange)
2	1k (brown black red)

3. Construction

1. Fit and solder all resistors, the diode, transistor and coil (break off case tabs).
2. Fit and solder all capacitors, noting polarity of electrolytics.
3. Fit and solder all links. Note that links A4, A5, A7 select the hardware address:

	LINK	register select	data I/O
(Normal)	A5	159	223
	A4	175	239
	A7	63	127

NB only 1 link to be connected: all software given assumes link A5.

3. Construction (continued)

4. Fit and solder 40 pin socket for AY-3-8910.

5. Fit and solder all the 74LS series integrated circuits, taking great care to observe orientation (see diagram).

6. Fit 23 way socket, if necessary breaking off the 2 unwanted (blanked) pins. Take great care to stand socket approximately ¼" off the printed circuit board (see diagram).

   Solder each pin very carefully to the main board.

7. Bend all socket pins inwards so that they can touch the stub board which is now soldered in place on each side.

   Take great care that the slot in the stub faces outwards, and matches the missing (blanked) connector pins. Solder all pins to the stub, top and bottom, very carefully. Check and double check for dry joints (try to pull the pins off!).

8. Insert the AY-3-8910 chip, observing correct orientation and taking care that no pins are accidentally buckled underneath.

4. Software

(Note: the routines described here are not to be entered when using the "Big Ears" Speech Recognition unit: refer to "Big Ears" user manual for all details).

The Stuart Synthesiser + I/O Port is designed as a true Z80 input/output device. This means that it is not memory mapped and, therefore, not directly accessible using PEEK and POKE commands. Conversely, since it does not occupy any memory addresses, it imposes no restrictions on system expansion beyond, say, 16K RAM.

Since Sinclair Basic does not provide any I/O instructions, it is necessary to enter a very small amount of machine code, which is called with the USR function.

Actually, Sinclair Basic is very weak in respect of its facilities for testing individual bits in a word or byte - essential when monitoring process inputs such as contacts or photocells - so we have provided some additional machine code routines to do this for you. (see attachment for Spectrum software)

Procedure for loading Machine Code Utilities: (ZX81)

Enter the Basic program given in lines 1 to 36. Note the large REM statement at line 1. This is a dummy location for the machine code, and should contain at least 100 characters - you can expand it later, as long as it remains line 1). To enter 100 characters, just type 1234567890 repeatedly 10 times.

/cont...

4. Software (continued) (N.B. For RETURN please read NEW LINE)
(ZX81)

Now type RUN. The screen should clear, and the question ADDRESS? should appear.

You are now in a memory examine/change (debug) routine, and can enter machine code from the listing in Table 1. Type in the first address, 16514, followed by RETURN, and the current contents will be displayed. Now type in the required contents from Table 1, followed by RETURN. The entry (change) will be confirmed on the screen, and the next location (16515) displayed. Carry on entering the new contents until the entire code has been typed in. Check periodically that the successive addresses tally with the left-hand column of Table 1, which gives every tenth address. If you make a mistake, or wish to check what you have done, then simply type a full stop (.) instead of new contents, followed by RETURN. This returns the programme to address-mode. When checking the contents, simply press RETURN to advance to the next location without making any change.

When the machine code has been entered and fully checked, type an X followed by RETURN to terminate the programme, then SAVE the programme on a new tape.

Keep this tape and use it to pre-load the computer with the machine code utilities when you are about to program a new application. (Note that lines 2-36, the loader program, can be deleted once the machine code has been entered).

Machine Code Utility Program (ZX81)

```
1 REM  12345678901234567890123456789012345678
90123456789012345678901234567890 . . . etc. (see text)
```

```
10 REM MACHINE CODE LOADER (C) 1981 WM STUART SYSTEMS
12 SCROLL
13 PRINT "ADDRESS?"
14 INPUT A$
16 IF A$ = "" THEN GOTO 20
18 LET AD = VAL (A$)
19 GOTO 21
20 LET AD = AD + 1
21 SCROLL
22 PRINT AD, PEEK (AD);
24 INPUT C$
26 IF C$ = "" THEN GOTO 20
28 IF C$ = "." THEN GOTO 12
30 LET CON  = VAL (C$)
32 POKE AD, CON
34 PRINT "/"; CON ;
36 GOTO 20
```


/cont...

Table 1 (2×81)

16514	0	0	0	58	130	64	211	159	58	131
16524	64	211	223	0	0	201	58	130	64	211
16534	159	219	223	6	0	79	201	205	146	64
16544	58	132	64	177	195	141	64	205	146	64
16554	58	132	64	47	161	195	141	64	205	146
16564	64	58	132	64	161	79	201			

5. Application Programming

Before trying to program the Synthesiser and I/O Port, it is necessary to understand the AY-3-8910 functions, shown as a block diagram in fig. 2.

The device contains 16 registers, each 8 bits wide, where each register serves to control a particular function.

Briefly: RØ to R5 control the frequencies of the 3 music channels.

R6 controls the character of an internal noise generator. R7 establishes, bit by bit, the mode of operation, i.e. whether Ports A & B are to behave as inputs or outputs, and whether tones and/or noise is enabled on each music channel. Thus

01111000 = 78 hex = 120 decimal

will set Port B to input mode, Port A to output mode, disable noise on all 3 channels and enable tones on all 3 channels. R8, 9 and 10 control the sound amplitude: if bit 4 ("M") = 0 then bits 0 to 3 ("L0 . . . L3") control amplitude directly in 15 steps (0 = min, 15 = max).

If bit 4 ("M") = 1 then bits 0 to 3 have no effect and the channel in question is controlled by an automatic envelope generator.

R11 and 12 control the automatic envelope period, i.e. a small value giving fast attack or decay.

R13 selects the type of envelope produced: see fig. 3 for graphic details of how these bits are used. For a classic "one shot" decaying sound envelope (percussive note, or explosion e.g.) set both "continue" and "hold" (1001 binary = 9 decimal). This envelope is generated each time the value is output to the register.

R14 and R15 represent the two input/output ports - in input mode the register is read to give the value represented by the 8 input signals, in output mode the register is written to in order to set the 8 output signals. Note that bits 6 and 7 of register 7 are set to determine which ports are in input mode, and which are in output mode.

1 = output
0 = input mode

/cont...

The 16 registers are accessed in 2 stages:

1. Register Select
2. Data Transfer: IN to Computer, OUT to Register.

Using the Machine Code Routines which you have already entered:

LET IN	=	16530	(see seperate sheet for Spectrum)
LET OUT	=	16517	
LET ON	=	16541	
LET OFF	=	16551	needed at the beginning
LET TEST	=	16562	of your programs
LET REG	=	16514	
LET CON	=	16515	
LET MASK	=	16516	

REGISTER \ BIT		B7	B6	B5	B4	B3	B2	B1	B0
R0	Channel A Tone Period	8 bit fine tune A							
R1						4 bit coarse			
R2	Channel B Tone Period	8 bit fine tune B							
R3						4 bit coarse			
R4	Channel C Tone Period	8 bit fine tune C							
R5						4 bit coarse			
R6	Noise Period					5 bit period			
R7	ENABLE	In/out		noise			tone		
		IOB	IOA	C	B	A	C	B	A
R8	Channel A Amplitude				M	L3	L2	L1	L0
R9	Channel B Amplitude				M	L3	L2	L1	L0
R10	Channel C Amplitude				M	L3	L2	L1	L0
R11	Envelope Period	8 bit fine tune E							
R12		8 bit coarse tune E							
R13	Envelope Shape					see fig 3			
R14	I/O Port A	8 bit parallel I/O Port A							
R15	I/O Port B	8 bit parallel I/O Port B							

Fig 2.

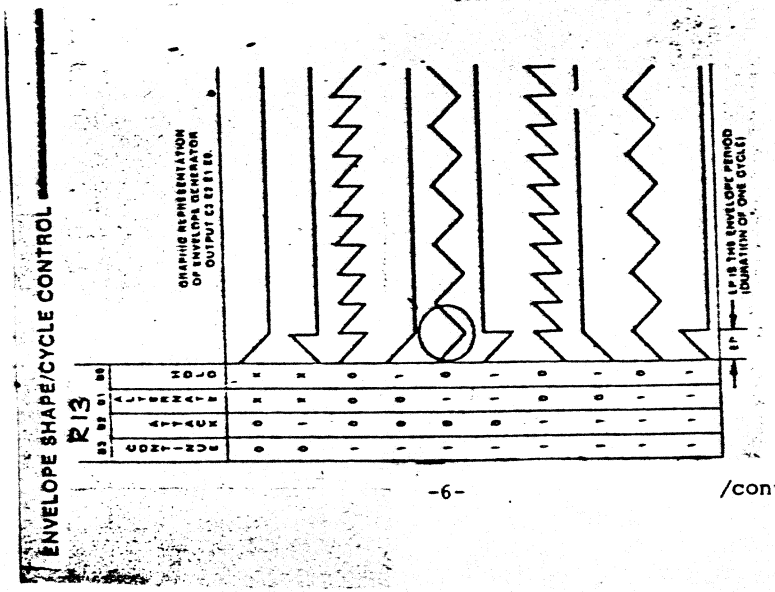


Fig 3.

/cont...

1. Register Selection

Example:

```
POKE REG, 7
          ↖ desired register, value 0 to 15.
```

(Note a register remains selected until another value is POKEd into REG).

2. Output to Register

(Select register as in 1. above).

Example:

```
POKE CON , 123
           ↖ desired value to be output, value 0 to 255.

LET X = USR (OUT)
```

3. Input from Register

(Select register as in 1. above).

Example:

```
LET X = USR (IN)
```

will set X to the value contained in the register - e.g. state of input port etc.

4. Selective Output

To set chosen bits within an 8 bit register, the following routines are used:

```
POKE REG, 7      defines chosen register
POKE MASK, 64    64 defines chosen bit
```

then `LET X = USR (ON)` (turns bit 6 on)

or `LET X = USR (OFF)` (turns bit 6 off)

bit:	7	6	5	4	3	2	1	0
value:	128	(64)	32	16	8	4	2	1

BIT VALUES

Note: the routines ON and OFF can be used to control more than one bit at a time, e.g.

```
POKE MASK, 7
LET X = USR (ON)
```

will turn on bits 0, 1 & 2, because 7 = 4 + 2 + 1 = combined values of the chosen bits.

Remember - the routine USR (OUT) forces all 8 bits to the value defined in CON . The routines USR (ON) and USR (OFF) permit independent control of those bits which are defined in MASK.

/cont...

5. Testing Individual Bits

Example: to test bit 4 of register 15

```
POKE REG, 15
POKE MASK, 16    selects bit 4
LET X = USR (TEST)
```

this sets X = 0 if the bit is clear
X ≠ 0 if the bit is set

If MASK is set to represent more than one bit, then
X = 0 if all chosen bits are zero
X ≠ 0 if any chosen bit is set.

6. Simple Example of Sound Generation

```
LET REG = . . .
LET
. . .                etc. as before

POKE REG, 7
POKE CON , 7
LET X = USR OUT          (enable Noise)
POKE REG, 6
POKE CON , 31
LET X = USR OUT          (set low freq. noise)
POKE REG, 12
POKE CON , 8
LET X = USR OUT          (set slow envelope period)
POKE REG, 13
POKE CON , 8
LET X = USR OUT          (set repeat envelope ∿∿∿)
POKE REG, 9
POKE CON , 16
LET X = USR OUT          (set channel B to envelope control)
```

This will give a slow steam train chuffing noise on channel B.

7. Hardware Connections

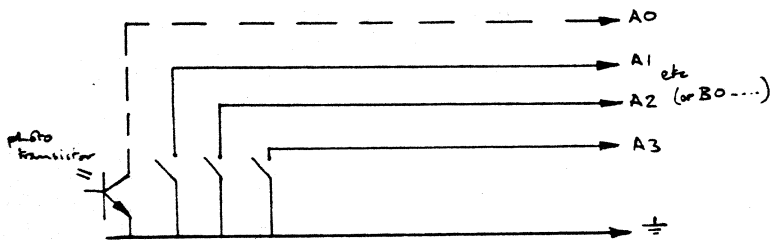
Connections are all made to the User Edge connector and are indicated in fig. 1. (If you do not wish to solder to your board, a suitable connector socket is available, price £3.00).

/cont...

The input/output ports A & B can be used as follows:

a) Input Mode

To sense the state of external switches, connect as follows:

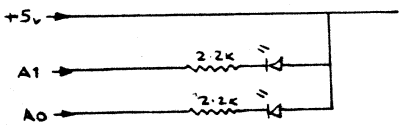


Note that when using input mode, the Stuart I/O board provides internal pull-up resistors, hence the extremely simple connection.

b) Output Mode

Outputs can sink (logic low) up to 1.6 mA, but source (logic high) only 100 uA.

Thus to drive l.e.d.s.

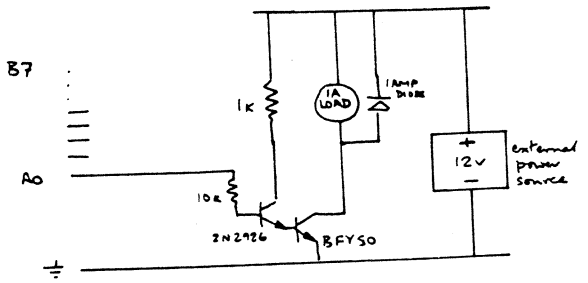


NB set bits = 0 to light l.e.d.s.

/cont...

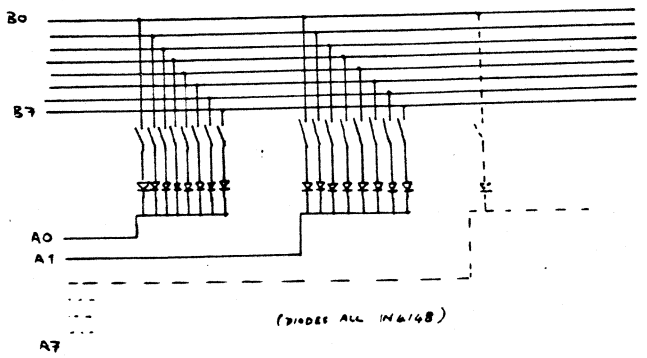
To drive greater loads, the following circuitry may be added:

Relays, Motors, etc.



c) Scanning Large Numbers of Inputs

Up to 64 inputs (key closures) may be scanned, using port A in output mode to address 8 contacts at a time. Port B reads the status in input mode.



Only one bit in Port A is set low at a time. This selects one group of 8 contacts and their status can be read with Port B.

Simple tune player for 1K systems. (N B. do not enter any other software.)

(ZX81)

a.) Enter and save this program: (do not try to run it yet!)

```
 1  REM  12345678901234567890123456?
 2  FAST
59  LET OUT = 16517
63  LET REG = 16514
64  LET CON = 16515

7Ø  LET ROOT = 17ØØ/256
72  POKE REG, 9
74  POKE CON, 16
75  LET X = USR OUT
77  POKE REG, 7
78  POKE CON, 61
79  LET X = USR OUT

9Ø  POKE REG,12
92  POKE CON,15
94  LET X= USR OUT
96  LET K= CODE (INKEY$)-27
98  IF K <1 THEN GOTO 96

100 IF K > 36 THEN GOTO 96
101 LET Q = ROOT / (2**((PEEK (16529+K)) /12))
102 LET H = INT Q
103 POKE REG, 2
104 POKE CON, INT ((Q-H)*256)
105 LET X = USR OUT
108 POKE REG, 3

110 POKE CON,H
112 LET X = USR OUT
114 POKE REG,13
116 POKE CON,9
118 LET X = USR OUT

120 IF INKEY$ <> " " THEN GOTO 12Ø
122 GOTO 96
```

*

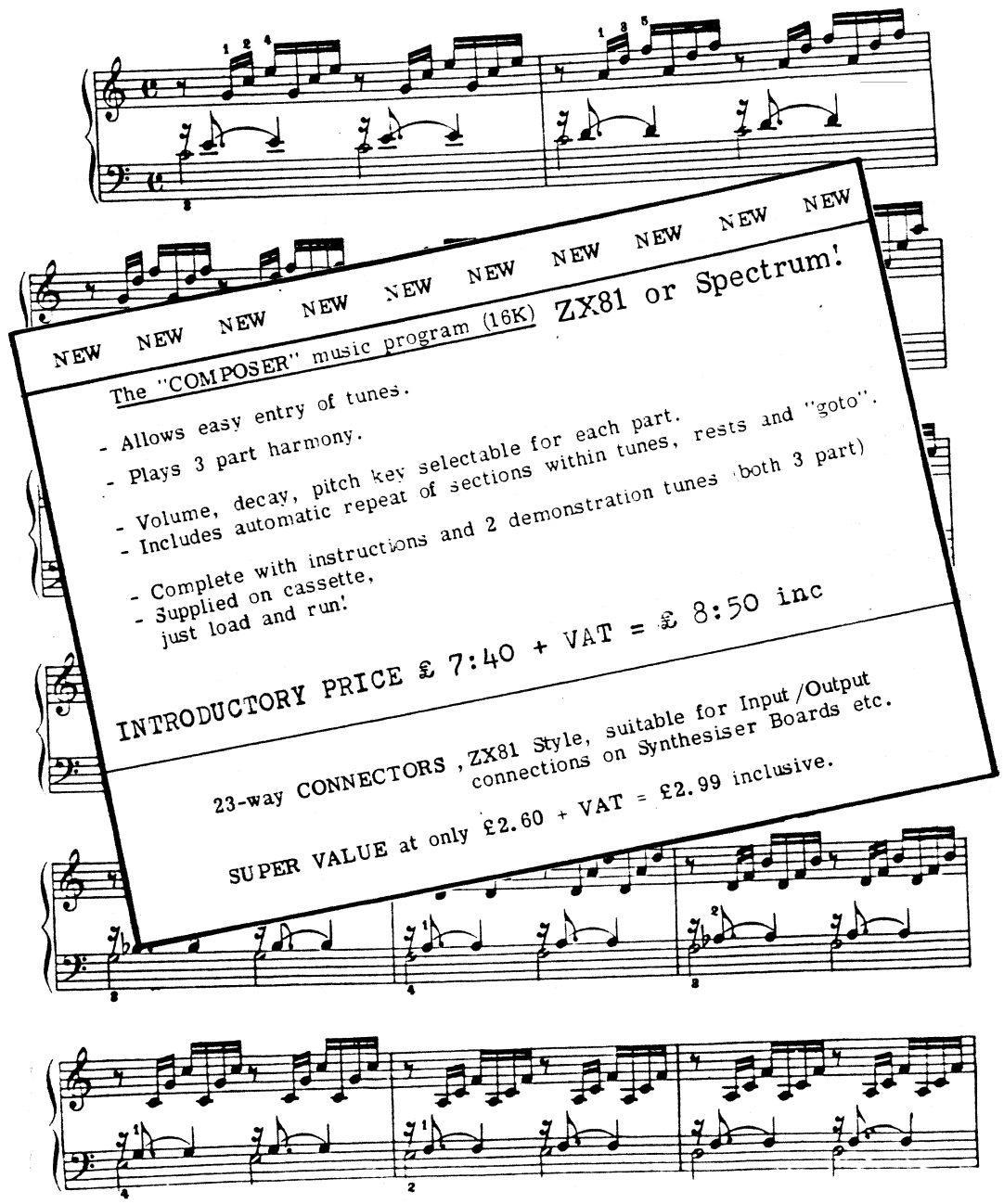
```
POKE 16517, 58
  "  16518,130
  "  16519, 64
  "  16520,211
  "  16521,135
  "  16522, 58
  "  16523,131
  "  16524, 64
  "  16525,211
  "  16526,199
  "  16527,  0
  "  16528,  0
  "  16529,201
  "  16530, 17
  "  16531,  1
  "  16532,  3
  "  16533,  5
  "  16534,  6
  "  16535,  8
  "  16536, 10
  "  16537, 12
  "  16538, 13
  "  16539, 15
```

b.) POKE all the values contained in the following list*, then SAVE the program again. (The REM statement will now contain strange looking characters)

c.) Type RUN — the top row of the keyboard plays tunes.

Standard Repair Charge for D.I.Y. Kits. £8:00 please.

Praeludium I.

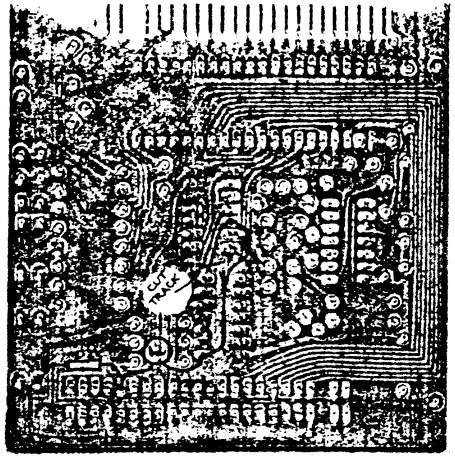


ERRATA

Mk III boards — Spectrum only

We regret that problems experienced with certain sound ICs on Spectrum, require the following modifications

NB track-side view



1. 2·2k resistor to be added from R6/R7 junction to Sinclair connector, top row, 5th pin from left as seen from track side.
2. 0·001μF (= 1nF = 1000pF) from same connector pin to IC2 pin 1. Disconnect for ZX81
3. Link IC2 pins 2, 3 and 4.
4. Cut track between IC2 pin 6 and IC3 pin 12.
5. Link IC3 pin 12 to IC2 pin 1

(NB steps 4 & 5 above are optional & only apply if Spectrum IN and OUT instructions are to be used).

We apologise for the inconvenience caused by these changes.

SPECTPUM I/O BOARD SOFTWAPE

Chapter 4 of the manual does not apply to Spectrum.

In order to load the routines for controlling the Music I/O Board, simply enter the following programme, and SAVE on tape.

```
  1 REM    SPECTRUM  I/O MkIII
  5 CLEAR 32499
 10 LET ad =32500
 12 LET cc=0
 20 READ n
 22 IF n<256 THEN GO TO 28
 24 IF cc=n THEN GO TO 50
 25 PRINT cc,"error in data": S
TOP
 28 POKE ad,n
 29 LET cc=cc+n
 30 LET ad=ad+1
 32 GO TO 20
 40 DATA 0,0,0,58,244,126,211,1
59,58,245,126,211,223,201
 45 DATA 58,244,126,211,159,219
,223,6,0,79,201
 46 DATA 205,4,127,58,246,126,1
77,195,255,126
 47 DATA 205,4,127,58,246,126,4
7,161,195,255,126
 48 DATA 205,4,127,58,246,126,1
61,79,201,7664
 50 LET reg=32500
 55 LET con=32501
 56 LET mask=32502
 60 LET out=32503
 65 LET in=32514
 66 LET on=32525
 67 LET off=32535
 68 LET test=32546
```

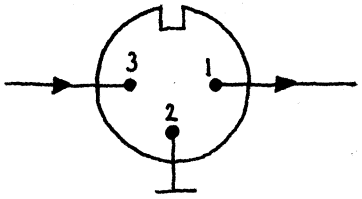
Now read the manual from Chapter 5 onwards.

The Simple Tune Player for Spectrum is shown below, as an addition to the above routines (i.e. enter lines 1-68 above)

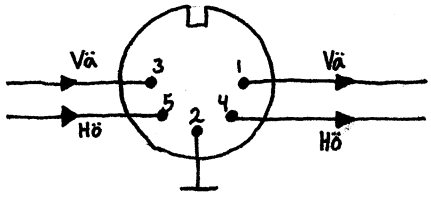
```
 70 LET root=1700/256
 72 POKE reg,9
 74 POKE con,16
 75 LET x=USR out
 77 POKE reg,7
 78 POKE con,61
 79 LET x=USR out
 80 DIM t(10)
 81 FOR n=1 TO 10: READ t(n)
 82 NEXT n
 84 DATA 17,1,3,5,6,8,10,12,13,
15
 90 POKE reg,12
 92 POKE con,15
 94 LET x=USR out
 96 LET k=CODE (INKEY$)-47
 98 IF k<1 THEN GO TO 96
100 IF k>10 THEN GO TO 96
101 LET q=root/(2↑(t(k)/12))
102 LET h=INT q
103 POKE reg,2
104 POKE con,INT ((q-h)*256)
105 LET x=USR out
108 POKE reg,3
110 POKE con,h
112 LET x=USR out
114 POKE reg,13
116 POKE con,9
118 LET x=USR out
120 IF INKEY$<>"" THEN GO TO 12
0
122 GO TO 96
```

DIN - kontakt

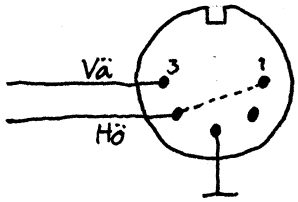
förstärkare



På 1-kanaliga förstärkare med 3-poliga DIN-kontakter kopplas anslutningarna på detta sätt.

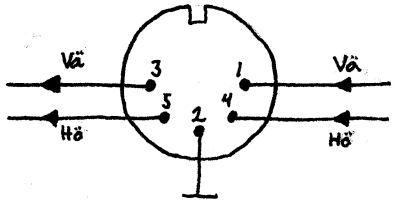


2-kanaliga förstärkare skall vara kopplade på detta sätt i DIN-uttagen.

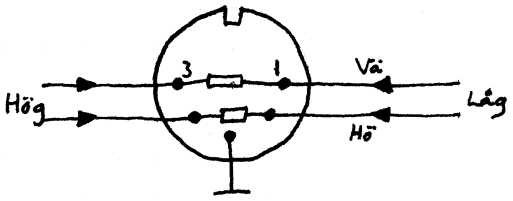


Pick up-ingången kan ibland vara utförd med överkoppling mellan stift 5 och 1 för att möjliggöra användning av 3-polig DIN-kontakt.

bandspelare

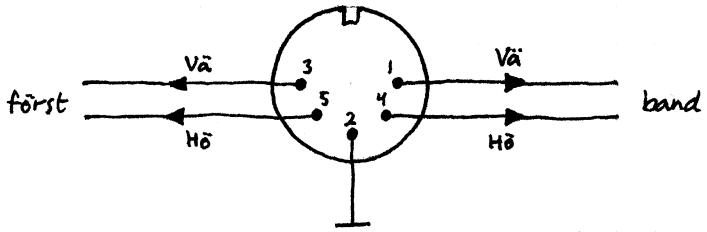


På bandspelare kopplas anslutningarna spegelvänt jämfört med förstärkaren. På så sätt vet man att alla stift skall anslutas till stift med motsvarande nummer på förstärkaren.



I vissa fall förekommer det att ut- och ingångarna kopplas samman genom dämpande resistanser hos bandspelare. Utgången fungerar då som högnivåingång vid inspelning.

tuner



På en tuner måste utgångarna till förstärkare resp bandspelare kopplas olika för att passa korrekt.

Följande tabell ger frekvenser (avrundade till heltal) för fyra oktaver med en halvtons mellanrum. Visserligen är inte listan fullständig för alla musiktoner, men den kan ändå vara en god hjälp vid programmering av musik.

Frekvens	Ton	Frekvens	Ton
110	A	440	A (över medel-C)
117	A#,B^b	466	A#,B^b
123	B	494	B
131	C (låga C)	523	C (höga C)
139	C#,D^b	554	C#,D^b
147	D	587	D
156	D#,E^b	622	D#,E^b
165	E	659	E
175	F	698	F
185	F#,G^b	740	F#,G^b
196	G	784	G
208	G#,B^b	831	G#,A^b
220	A (under medel-C)	880	A (över höga C)
233	A#,B^b	932	A#,B^b
247	B	988	B
262	C (medel-C)	1047	C
277	C#,D^b	1109	C#,D^b
294	D	1175	D
311	D#,E^b	1245	D#,E^b
330	E	1319	E
349	F	1397	F
370	F#,G^b	1480	F#,G^b
392	G	1568	G
415	G#,A^b	1661	G#,A^b
		1760	A

Compiled by DJD.

Appearing every two months, Micro-Bus presents ideas, applications, and programs for the most popular microprocessors; ones that you are unlikely to find in the manufacturers' data. The most original ideas often come from readers working on their own systems; payment will be made for any contribution featured.

THIS month's Micro-Bus features an EPROM extension board for the ZX81 or ZX80, a BASIC program to solve simultaneous equations, and the solutions to the BASIC problems featured in the last Micro-Bus.

EPROM EXTENSION BOARD

The circuit shown in Fig. 1 was submitted by *I. P. Bryant*, and will add up to 8K of EPROM or RAM to a ZX80 or ZX81. The extra memory appears in the memory space directly above the 8K monitor ROM, and the circuit uses 2716 or 2516 EPROMs as these are available at a reasonable price.

The circuit is interfaced to the ZX81 bus by means of the edge connector. For ZX80 users there is no ROM chip select signal on the connector, but this can be created by connecting IC6 pin 6 to the spare edge connector, and then cutting the track between IC12 pin 11 and IC6 pin 6 and connecting a 680 ohm resistor across the break. The edge connector will now be identical to the ZX81's.

The circuit does not need to include any write protection because the 1k resistors in the ZX80/81's data bus provide this.

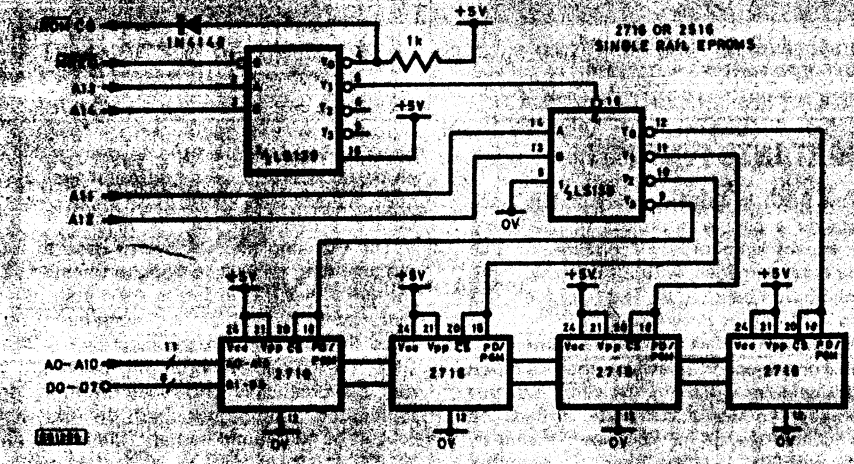


Fig. 1. Circuit adds 8K of EPROMs to a ZX80 or ZX81

SIMULTANEOUS EQUATIONS

Simultaneous equations crop up in many different branches of electronics. For example, one can determine the resistive, capacitive, and inductive components of a passive component by measuring its impedance at three different frequencies, and then solving three simultaneous equations to find R, C, and L. The BASIC program of Fig. 2 was submitted by *A. Schoultz* of South Africa, and can be used to solve such equations. The program was developed on a ZX81, but can fairly easily be modified to run on any BASIC-speaking microcomputer.

The number of equations which can be solved is dependent only on the amount of available memory. On a ZX81 with a 16K RAM pack the program has no trouble solving ten equations with ten unknowns, and takes about 22 seconds. As an example, to solve the three equations:

$a + 2b + 3c = 42$
$7a + b - 8c = 36$
$3a - 2b + 2c = 54$

one enters the coefficients in order:
1, 2, 3, 42, 7, 1, −8, 36, 3, −2, 2, 54.
The program should then produce the correct answer:
$a = 14, b = 2, c = 8$.

PROGRAM OPERATION

The program, lines 10 to 150, first inputs the number of equations and sets up three arrays. Array A holds the coefficients of the variables, and array B holds the constants. The solution is returned in array X. Lines 1000 to 1036 solve the equations, and lines 1037 to 1048 print out the solution.

To alter the program for other machines, those features peculiar to the ZX81 will have to be altered. The value of Q on line 15 is chosen so that CHR$(Q+1) will give the letter A, for the printing of the equation variables; other machines that use ASCII will need Q=64. The PRINT AT Y,X statement in lines 110, 140, 1043, and 1046 can be replaced by

```
  5 REM *** SIMULTANEOUS EQUATIONS ***
 10 PRINT "NO. OF EQUATIONS =";
 15 Q=37
 20 INPUT N
 30 PRINT N
 40 DIM A(N,N)
 50 DIM B(N)
 60 DIM X(N)
 70 LET C=INT(30/(N+1))
 80 FOR I=1 TO N
 90 FOR J=1 TO N
100 INPUT A(I,J)
110 PRINT AT I*2,J*C-C;A(I,J);" ";CHR$(J+Q)
120 NEXT J
130 INPUT B(I)
140 PRINT AT I*2,J*C-C;"= ";B(I)
150 NEXT I

1000 FAST
1001 FOR I=1 TO N-1
1002 LET R=I
1003 LET M=A(I,I)
1004 FOR J=I+1 TO N
1005 IF A(J,I)<=M THEN GOTO 1008
1006 LET R=J
1007 LET M=A(I,J)
1008 NEXT J
1009 IF M=0 THEN PRINT "SOLUTION ABORTED."
1010 IF R=I THEN GOTO 1019
1011 FOR K=1 TO N
1012 LET S=A(I,K)
1013 LET A(I,K)=A(R,K)
1014 LET A(R,K)=S
1015 NEXT K
1016 LET S=B(I)
1017 LET B(I)=B(R)
1018 LET B(R)=S
1019 FOR J=I+1 TO N
1020 LET N=A(J,I)/A(I,I)
1021 FOR K=1 TO N
1022 LET A(J,K)=A(J,K)-N*A(I,K)
1023 NEXT K
1024 LET B(J)=B(J)-N*B(I)
1025 NEXT J
1026 NEXT I
1027 REM BACK SUBSTITUTION
1028 LET X(N)=B(N)/A(N,N)
1029 FOR I=N-1 TO 1 STEP -1
1030 LET S=0
1031 FOR J=I+1 TO N
1032 LET S=S+X(J)*A(I,J)
1033 NEXT J
1034 LET X(I)=(B(I)-S)/A(I,I)
1035 NEXT I
1036 PRINT "-----------------------------"
1037 SLOW
1038 FOR I=1 TO N
1039 LET J=(N+1+1)*2
1040 IF J<=20 THEN GOTO 1046
1041 SCROLL
1042 SCROLL
1043 PRINT AT 20,0;CHR$ (I+Q);" = ";X(I)
1044 NEXT I
1045 STOP
1046 PRINT AT J,0;CHR$ (I+Q);" = ";X(I)
1047 NEXT I
1048 STOP
```

Fig. 2. Program for the ZX81 will solve simultaneous equations

MICRO-BUS

Compiled by DJD.

Appearing every two months, Micro-Bus presents ideas, applications, and programs for the most popular microprocessors; ones that you are unlikely to find in the manufacturers' data. The most original ideas often come from readers working on their own systems; payment will be made for any contribution featured.

THIS month's Micro-Bus focuses attention on the Sinclair ZX81 and ZX80 microcomputers, and includes two memory-expansion circuits, an encryption system, and a program renumber routine.

EXTRA RAM FOR ZX81

The programming space available on the ZX81 can be increased some 3 to 4 times by adding an extra 1K of memory, as shown in the simple circuit of Fig. 1. This was submitted by *D. Brownlee* of Hatfield who writes: "When I first took delivery of my ZX81 I attempted to enter a fairly long program published for the standard ZX80. However, I soon discovered that the program would not fit in, presumably because the ZX81 uses more of the RAM for its workspace. I overcame this problem with the RAM extension, and the circuit should be self-explanatory. The address decoding is performed by a 74LS10, and when the extra memory is addressed the internal memory is disabled by an extra gate and a diode."

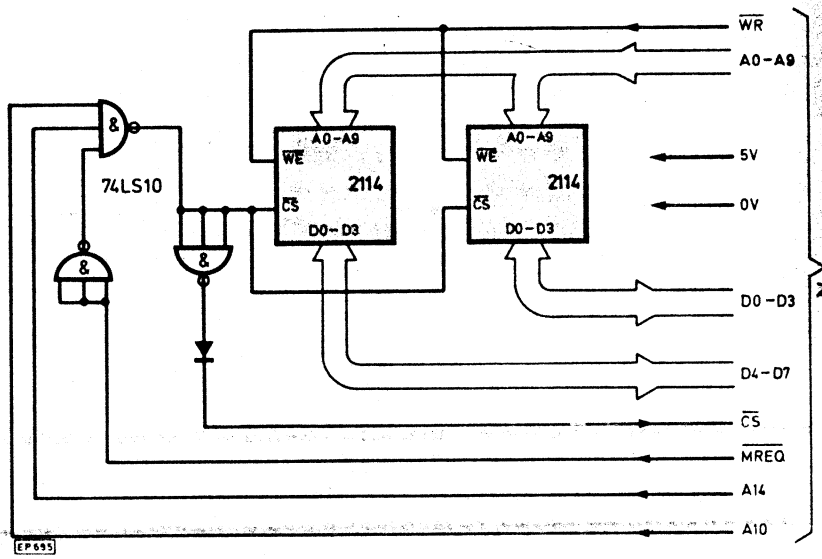


Fig. 1. 1K RAM Extension for a ZX81

7K EXPANSION

If more than 1K of expansion is envisaged the following design may be more suitable. It was sent in by *Howard Parry* of Manchester, and can accommodate up to 7K of RAM, or a mixture of RAM and I/O ports; what follows is based on his description.

"For the expansion I decided to use 2114 static RAM chips, the same as those used in the kit. One problem was the edge connector; I could not find a 23 way double sided connector in any of the popular magazines, so I finally decided on a Vero double sided 43 way connector which I cut down to size. It fits perfectly!

EXPANSION CIRCUIT

"The circuit, shown in Fig. 2, is based around a 74LS138 decoder i.c., and the memory chips. It works as follows: the 74LS138 decodes A14 and $\overline{\text{MREQ}}$ to enable the expansion to be positioned at the start of the 16K area. A10-A11 and A12-A13 decode one of eight 1K blocks from 16K to 23K. $\overline{\text{MREQ}}$ is required to make sure that the address lines are stable before memory is accessed for read and write operations. To disable the internal 1K RAM block Y0 is connected back to $\overline{\text{CS}}$ on the ZX81 bus.

"After building the RAM expansion I did the RAMTOP test: PRINT PEEK 16388+256*PEEK 16389 and, hey presto, the answer came up 18432!"

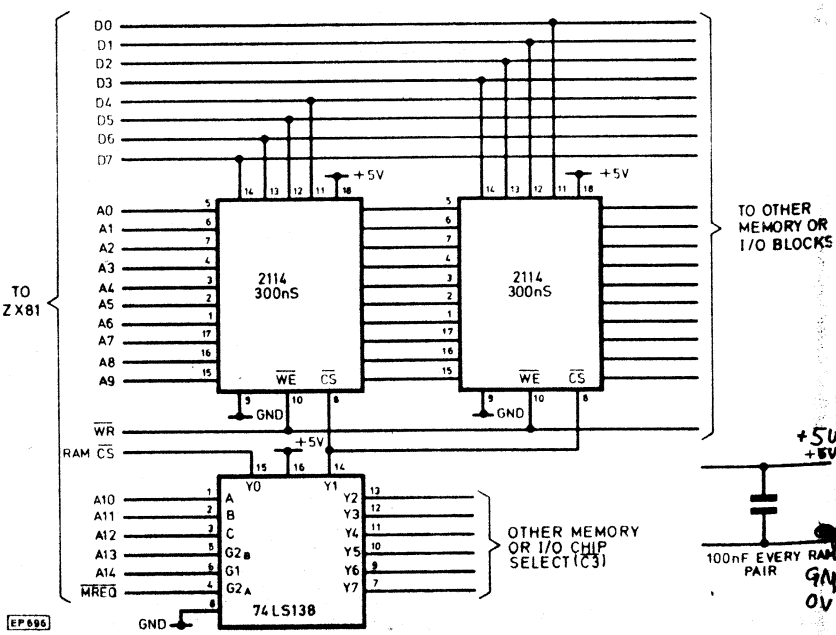


Fig. 2. Extension for the ZX81 can take up to 7K of RAM and/or I/O

and MICROPROMPT

WARM START FOR ZX81

Sir—One of the most frustrating features of the ZX81 is its lack of a hardware 'BREAK' key. This is particularly apparent when developing and trying out machine code programs: So long as the machine is in BASIC it is possible to exit from a program loop by means of the software-scanned BREAK key on the keyboard; but once the ZX81 ceases to scan the keyboard, as will occur during the execution of a machine code program which either intentionally or unintentionally gets stuck in a loop, one loses all control of the machine; it becomes completely inaccessible and unresponsive. All you can do is pull the plug on it, reconnect, re-load your programs, and start all over again.

I have tried many different approaches to try to implement a Warm Start, and finally came up with a successful method which is extremely simple.

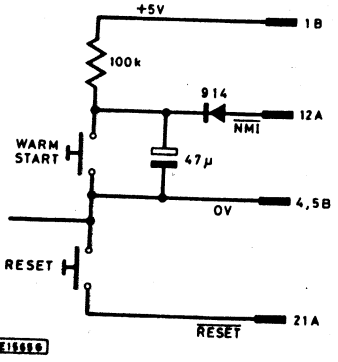
The obvious approach is to use the interrupt system, but I was initially discouraged from doing this because the ZX81's interrupt system is completely tied up with its display hardware and routines. Furthermore, the interrupt 'vectors' for the NMI and all INT modes except 2 (fully vectored interrupts) lie in the ROM, and so cannot be changed. On receipt of an NMI, control is transferred to the frame-configuring routine at 0066 hex; a mode 0 INT transfers control to the line-outputting routine at 0038 hex.

However, believe it or not, there is a way. Inspection of the NMI handler routine reveals that it is normally exited by either a RET or by a JP (IX), an indirect jump to the address pointed to by the IX register. Now, provided that the alternative accumulator contains zero, after the initial incrementing at the beginning of the NMI routine, exit from the routine will be via the JP (IX). Thus all that is necessary is to preface one's machine code program with a segment of "rescue code" which ensures that the alternative accumulator will be found to contain zero after its initial incrementing by the NMI routine, and that the IX register will contain an appropriate address for re-entry into the BASIC command-level keyboard scan/display loop. Now if an NMI pulse is generated (the $\overline{NMI}$ input of the Z80 is negative edge sensitive), control will be transferred to the NMI routine at 0066 hex in ROM, and an indirect jump will be forced back into BASIC.

There are two very important additional points: (1) This method is only possible in FAST mode, when the NMI routine is not in use for display purposes, and the SCL chip is not generating $\overline{NMI}$ pulses. This is no problem, because chances are one will want to execute the machine code program without interruption by the system display hardware (except of course for flicker-free graphics applications), and once back in BASIC one can switch between FAST and SLOW modes as desired, (2) The Warm Start push button must not be used while in BASIC, otherwise the system will hang up.

Clearly it is also necessary to avoid using the IX register and the alternative accumulator for any other purpose.

Thus it is necessary only to make sure that the system is in FAST mode before doing the USR call, and to make sure that the Warm Start button is used only to exit from a machine code program which has been prefaced by the rescue code. I have found that the easiest way—in conjunction with the ZXAS Assembler by Bug-Byte—is to LOAD "ZXAS", RUN, NEW, and then LOAD a "starter" program which incorporates an initial REM line with enough space for machine code, and the rescue code assembly listing, and the BASIC part of



ZXAS needed to actually use it (slightly modified—to ensure that FAST mode is retained throughout).

Once the rescue code has been assembled from decimal 16514 in the initial REM line, the source listing for the rescue code may, if desired, be deleted. Thereafter assembly should be from location dec. 16521, which is the next location after the rescue code. It will be found that it is possible to do a Warm Start out of the machine-code routine by a single press of the Warm Start button. The rescue code may of course be made into a separate subroutine, useful when working with more than one machine code program, or with a program with multiple entry points.

Note that a USR call from BASIC automatically disables maskable interrupts, so your machine code program will not be interrupted by the INT handler at 0038 hex (unless you should explicitly re-enable interrupts in the course of your code), and on encountering RET, or via the Warm Start via JP (IX) to address 0410 hex, maskable interrupts are re-enabled, so that the display system continues to work correctly.

The "Starter" Program for use with ZXAS. Certain lines are part of Bug-Byte's ZXAS Assembler, which should have been previously LOADed, followed by RUN, then NEW

```
10 REM 000000000000000000 . . . . . 0000
i.e. enough space for machine code
20 REM (
30 REM EX AF.AF'   }
40 REM LD A.255    }
50 REM EX AF.AF'   }  Rescue code source
60 REM LD IX.$0410 }  listing.

        Space for rest of assembly listing.

4000 REM )

        Space for additional BASIC programs.

9000 FAST
9010 INPUT ZZZ
9020 POKE 32461, INT (ZZZ/256)
9030 POKE 32460, ZZZ-256*INT
(ZZZ/256)
9040 RAND USR 28565
9050 PRINT AT 21,0;"ERROR"; PEEK
32651
```

For assembly use GOTO 9000. Assemble and execute from dec. 16514; if lines 30 through 60 deleted after assembling rescue code, then assemble program proper from dec. 16521, but still execute from dec. 16514. Rescue code in hex is 08 3EFF 08 DD211004.

So few components are required that they may easily be fitted to the ZX81 pcb. The push-to-make Warm Start switch may be fitted to the case or brought out to some convenient position. While carrying out this modification it is worth fitting a RESET switch at the same time; this achieves the same as the power-on reset, but without the need to disconnect or turn off the power supply.

Philip Creighton,
Luton,
Beds.

Spectrum Speech Synthesiser & 8 Bit I/O Port

G.Hodgson

Get your Spectrum/ZX81 thinking out loud for around £18

THERE have recently been a few designs published utilising the SPO256AL2 speech processor manufactured by *General Instruments*. However, this circuit, as well as incorporating an SPO256AL2, contains a Z80A PIO of which one port, port A, is dedicated to the speech i.c. and the other port, port B, is taken to a 15-pin "D" socket and so can be used as a programmable, bidirectional 8-bit I/O port.

The circuit has a very low component count using only two i.c.s, a regulator plus some other discreet components. The whole unit can be made for less than £18.

THE SPO256AL2 AND Z80A PIO

The SPO256AL2 is one of a family of speech processors produced by GI. It contains a clock circuit, micro-controller, 16K ROM, and digital on-chip filter making it a compact unit. The ROM contains the data required to produce 59 different sounds, or allophones, plus five pauses of different lengths. The required allophone is selected by placing a 6-bit address on A0 to A5 and then taking $\overline{ALD}$, allophone load, low when the data is latched into the i.c.

The Z80A PIO contains two bidirectional 8-bit ports, each of which is individually programmable. The i.c. is selected by taking pin 4, chip enable ($\overline{CE}$), low and then the relevant port register is selected using pins 5 and 6, C/$\overline{D}$ and B/$\overline{A}$ respectively, thus each port is latched. The i.c. has four operating modes for each port and port B is capable of driving Darlington transistors (1·5mA at 1·5V).

CIRCUIT DESCRIPTION

No discrete logic is required for address decoding since the PIO's internal logic will detect when it has been selected. Address line A7 is used for device selection and must be taken low for the PIO to be selected. A5 is connected to B/$\overline{A}$ and A6 to C/$\overline{D}$. Thus A5 and A6 are used for register selection:

A5	A6	A7	Port		FUNCTION
(B/A)	(C/D)	(CE)	decimal	hex	
0	0	0	31	1F	A Data
0	1	0	95	5F	A Control
1	0	0	63	3F	B Data
1	1	0	127	7F	B Control

Table 1. The PIO addressing details showing register/ port allocations

A0 to A4 must always be high since these lines are used to select Sinclair peripherals; a logic low on one of these lines will select a certain peripheral and, if that peripheral is attached, this would lead to two peripherals trying to use the system bus simultaneously.

$\overline{M1}$, $\overline{IORQ}$ and $\overline{RD}$ are used by the PIO to detect instruction type. INT is the interrupt output from the PIO; an interrupt is generated by certain conditions on either port A or port B, more later. $\overline{M1}$ and $\overline{IORQ}$ are used to detect whether an interrupt has been acknowledged since they will both go low to indicate an interrupt acknowledge. CLK is the standard Z80 single-phase clock taken from the host computer. Pins 22 and 24, IEO and IEI respectively, are used for "daisy-chaining" Z80 peripherals. In this circuit the PIO is configured as having highest priority since its IEI pin is taken to +5V via R1. When the PIO, IC1, is having an interrupt serviced, its IEO pin will go low preventing any lower priority device from generating an interrupt. Port B, consisting of B0 to B7, $\overline{BSTB}$, BRDY, is taken to SK2, a 15-pin female D-type socket. In this circuit $\overline{ASTB}$ and ARDY are not used since Port A is operated in Mode 3 only.

A0 to A5, READY and $\overline{ALD}$ from IC2 are taken to A0 to A7 on IC1 and so complete control over IC2 is exercised by port A, IC1. The two reset lines of IC2, pins 2 and 25, are taken to the Z80's RESET line and so IC2 is reset whenever the host computer is reset. PB1 is provided to generate a reset pulse whenever it is pressed since both the *ZX81* and *ZX Spectrum* lack a reset button. IC2's internal clock is used in conjunction with X1, a 2·4576MHz crystal, to provide the clock frequency for IC2. Varying this frequency, by inserting a different crystal in the range 1MHz–3·5MHz, will alter the speed of the speech. The output from IC2, pin 24, consists of a pulse-width modulated signal which is then passed through a 5·3kHz low-pass filter formed by R2, R3 and C5, C6. This filter converts the signal into an analogue one.

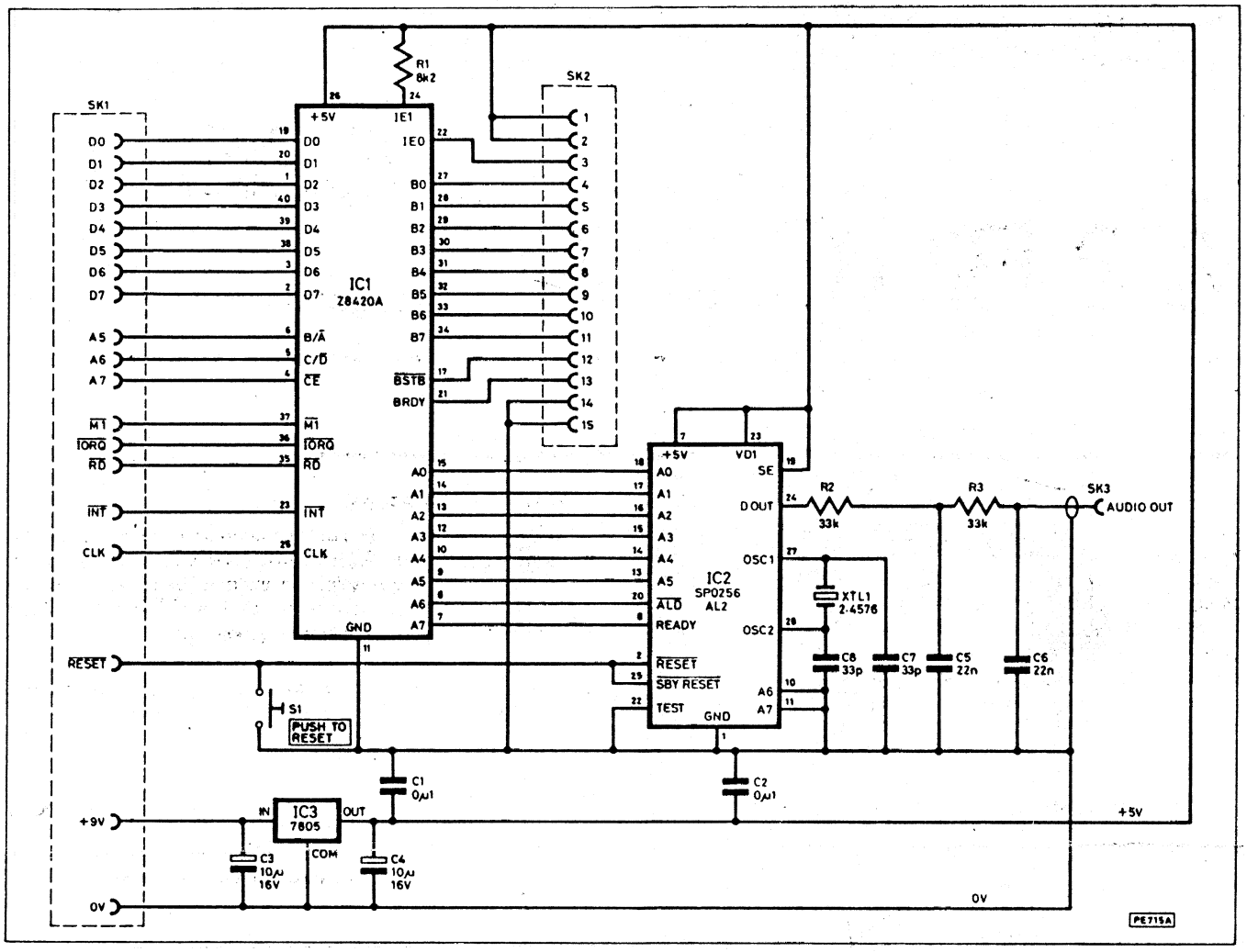


Fig. 1. Complete circuit diagram of the Spectrum, Speech Synthesiser and I/O Port

Further loading of the 7805 in either the *ZX81* or *ZX Spectrum* is not advised and so the 9V output from the power supply is fed into a 7805 regulator, IC3, in this circuit to provide power for the circuit.

C1 should be mounted as close as possible to IC1 and C2 should be mounted as close as possible to IC2 as these are decoupling capacitors, across the supply rails. SK1 is a 2 by 23-way 0·1 inch pitch edge connector.

CONNECTIONS USED

Reset connections are different for each computer and a small wire link or switch should be incorporated to get around this problem. All other connections are the same.

PROGRAMMING

Although IC1 can have each port programmed in four different modes, it has been found that mode 3 (control mode) is the most useful and, indeed, the easiest and so I shall just describe this mode. The port strobe and ready lines (ASTB $\overline{\text{BSTB}}$ ARDY BRDY) are internally inhibited in this mode but $\overline{\text{BSTB}}$ and BRDY have been made available for those who have a knowledge of modes 0, 1 and 2. The *Zilog* publication *Z80 PIO Technical Manual* describes fully the PIO and how to program it, for those who are interested.

In mode 3, each bit can be individually defined as either input or output. For interrupt control in this mode, the CPU interrupt mode must be mode 2, i.e. a machine code IM2 instruction should be executed followed by an EI instruction to enable the interrupts. If interrupts are not be be used then programming of the PIO can be accomplished quite easily.

COMPONENTS . . .

Resistors
R1 8k2
R2,R3 33k (2 off)
All resistors ¼W 5%

Capacitors
C1,C2 100n disc ceramic (2 off)
C3,C4 10µ 16V electrolytic (2 off)
C5,C6 22n polyester (2 off)
C7,C8 33p ceramic (2 off)

Semiconductors
IC1 Z8420A (Z80A PIO)
IC2 SPO256AL2
IC3 7805

Miscellaneous
X1 2·4576MHz crystal
S1 push-to-make switch
SK1 2 by 23-way 0·1" edge connector
SK2 16-pin female D-type socket
SK3 Phono socket
Printed circuit board PE–023 available from the PE PCB Service; suitable heatsink for IC3

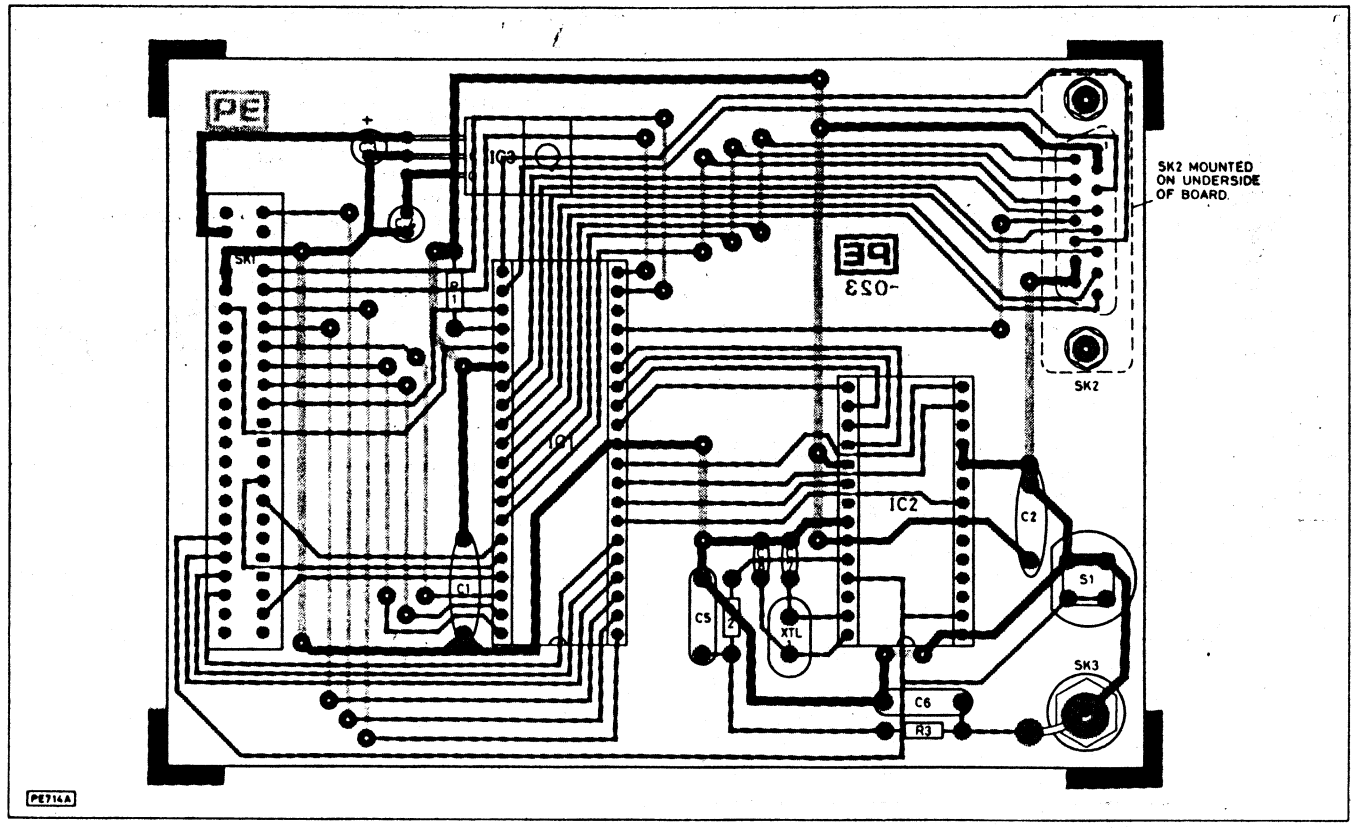


Fig. 2. P.c.b. design and layout

PIO PROGRAMMING

Port A

OUT 95, 255	Selects mode 3 for port A
OUT 95, x	x defines input and output bits: 0 sets bit to output, 1 sets bit to input, e.g. 255 would set all bits to input

Port B

OUT 127, 255	Selects mode 3 for port B
OUT 127, x	x defines input and output bits: 0 sets bit to output, 1 sets bit to input; e.g.

77 =	D7	D6	D5	D4	D3	D2	D1	D0
	O/P	1/P	O/P	O/P	1/P	1/P	O/P	1/P
	0	1	0	0	1	1	0	1

As interrupt mode 2 is rather hard to implement on a *ZX81* or *ZX Spectrum* it is not practicable to use PIO interrupts. Nevertheless, it can be done and that is why the PIO $\overline{INT}$ line is connected.

Programming the SPO256AL2 is very straightforward. When READY is high the i.c. will accept an allophone and so to send an allophone to the SPO256AL2 a suitable program can easily be constructed.

SOFTWARE

This is a small sample of software for driving the SPO256AL2; and is of a skeletal nature. The programs may be expanded, of course.

ZX SPECTRUM

```
LD A,255
OUT (95), A          Port A mode 3
LD A, 128
OUT (95), A          A0 to A6 output, A7 input
IN A, (31) (LP1)
BIT 7, A             Is READY high?
JR Z, (LP1)          Jump back if READY low
LD A, (23728)        Fetch allophone code
RES 6,A              ALD must be low
OUT (31), A          Send allophone
SET 6,A              Take ALD back high again
OUT (31), A
RET                  Back to BASIC
```

The allophone to be sent should be POKEd into 23728 which is an unused location in the system variables.

A BASIC program to poke the machine code into RAM is:

```
10 clear n           (decide on a value for RAMTOP)
20 FOR a=n+1 TO n+ 26
30 READ b: POKE a,b: NEXT a
40 DATA 62, 255, 211, 95, 62, 128, 211, 95, 219, 31, 203,
   127, 40, -6, 58, 176, 92, 203, 183, 211, 31, 203, 247,
   211, 31, 201.
```

An allophone can then be sent using:

```
POKE 23728,x
RANDOMIZE USR (n+1)
```

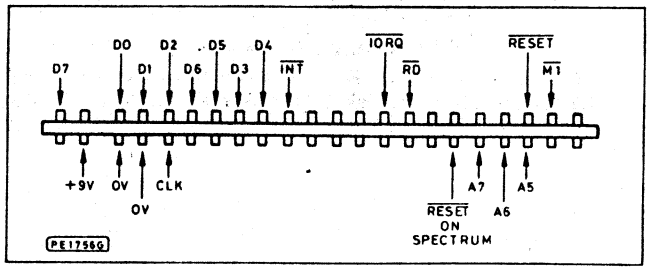


Fig. 3. The Spectrum and ZX81 edge connector

ZX81

The machine code program is the same for the *ZX81* as that for the *ZX Spectrum;* however, location 16507 is used to hold the allophone code. Again this is an unused location in the system variables. Thus the instruction LD A, (23728) becomes LD A, (16507) [5812364].

The basic program is different since READ? DATA and CLEARn are not available on the *ZX81*:

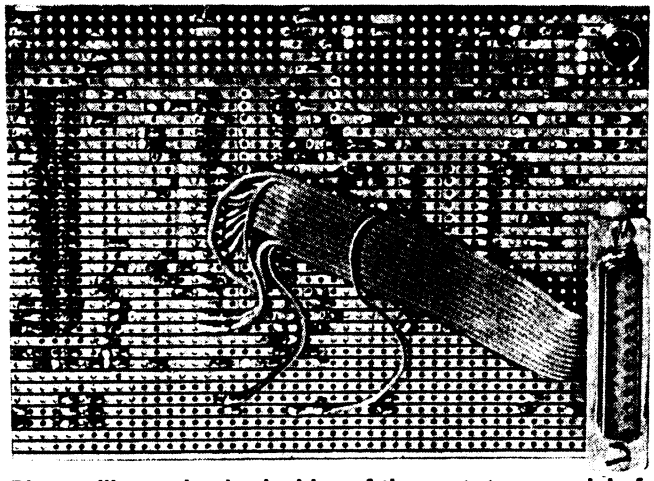
```
 10 REM XXXXXXXXXXXXXXXXXXXXXXXXXX
(26 X's)
 20 FOR A = 16514 TO 16539
 30 INPUT B
 40 POKE A, B
 50 NEXT A
```

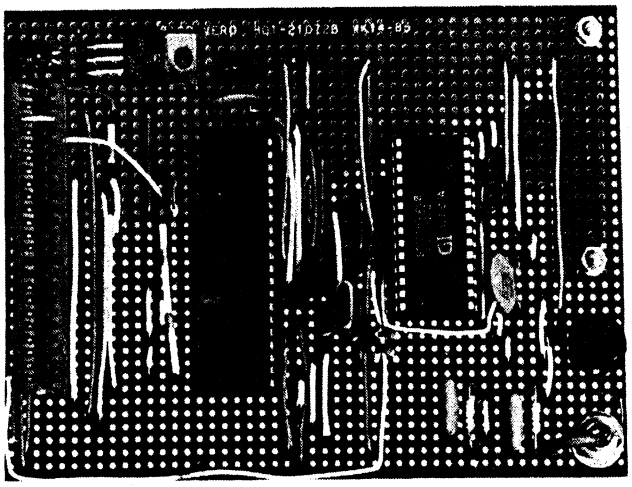
The above program should be RUN and the following 26 bytes entered.

```
 62   255
211    95
 62   128
211    95
219    31
203   127
 40   250
 58   123
203   183
211    31
203   247
211    31
201
```

When this data has been entered, lines 20–50 may be deleted but line 10 must be kept since it is used to hold the machine code. Note also that line 10 must be the first line in ANY program.

64 An allophone can then be sent using:
POKE 16507, x
RAND USR 16514





Photos illustrating both sides of the prototype model of the Spectrum Speech Synthesiser and 8-bit I/O Port. The prototype was built on Veroboard but the construction is similar to the p.c.b. design

CONSTRUCTION

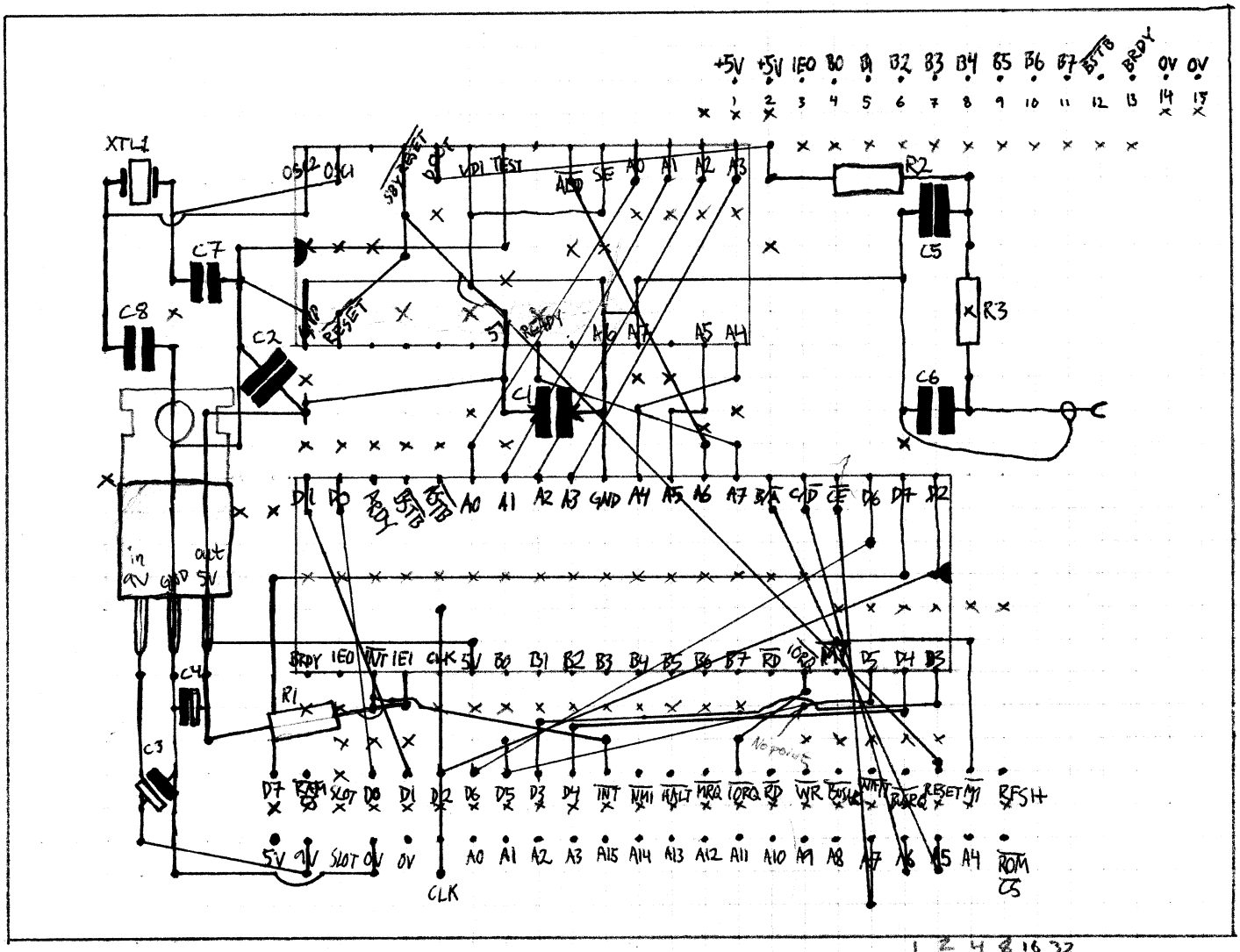
The prototype model of the Spectrum Speech Synthesiser and I/O Port was built on a small Veroboard as shown in the photographs. Because we think that this project will be very popular, we have designed a suitable p.c.b. of similar size to the prototype which is shown in Fig. 2.

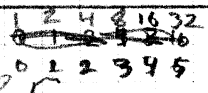
Construction should start by assembling the components on the p.c.b. in the normal way, resistors and capacitors first. Next, the i.c. sockets and the regulator should be mounted, together with the switch and the crystal.

Provision has been made on the p.c.b. for the D-type connector to be mounted directly on to the p.c.b. or wired accordingly. When all the components have been mounted, all solder joints and connections should be carefully checked for solder splashes and dry joints. Finally, the two d.i.l. i.c.s should be fitted in to their sockets making sure they are the right way round.

The project should be carefully connected to the computer edge connector (Fig. 3) and the Speech Synth And I/O Port may be used as described previously. You will probably find that further development of the computer software will enable better results. ★

Den verkliga layouten





SpeechSynthesiser

SpeechSynth

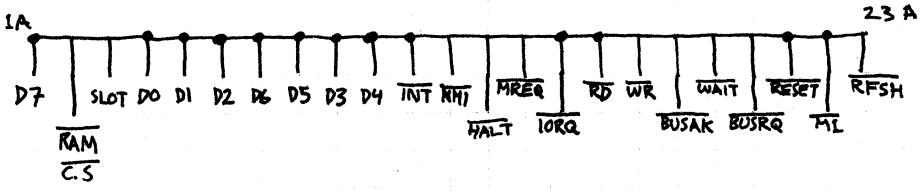
```
    ld a, 255
    out 95, a
    ld a, 128
    out 95, a
LP1: in a, 31
    bit 7, a
    jr nz, LP1
    ld a, (16507)
    res 6, a
    out 31, a
    set 6, a
    out 31, a
    ret
```

$2^5 = 32$
$2^6 = 64$
$2^7 = 128$

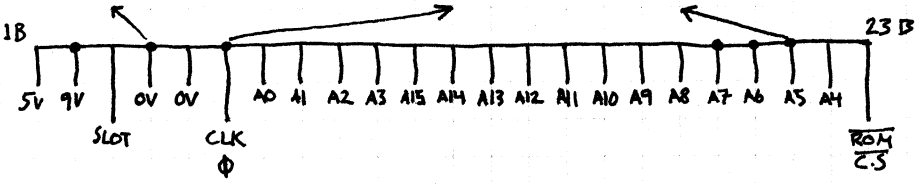
PIN-förteckning

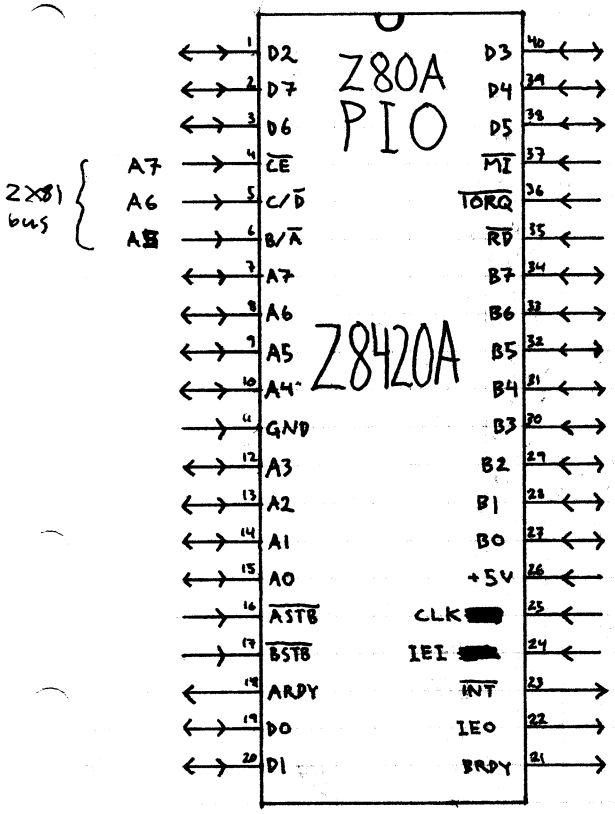
ZX81 CPU-buss

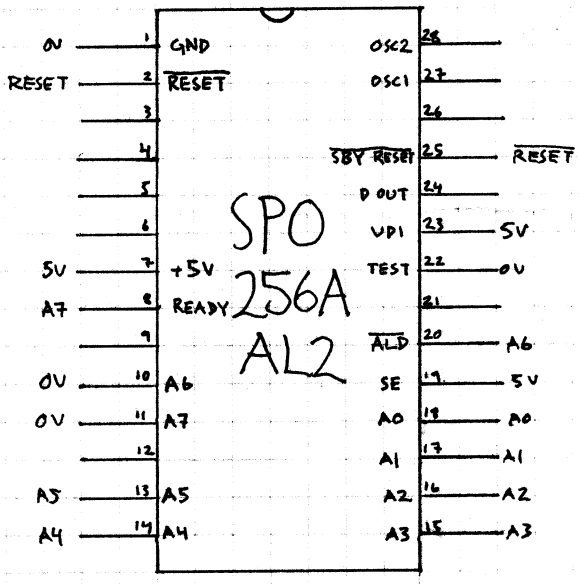
component side

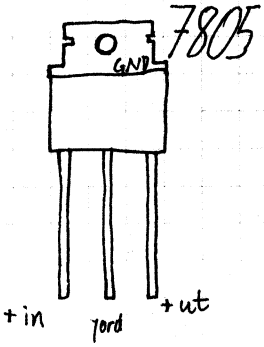


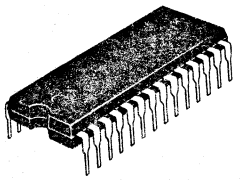
under side











MITSUBISHI LSIs

M5M5165P-70, -10, -12, -70L, -10L, -12L

65536-BIT (8192 WORD BY 8-BIT) CMOS STATIC RAM

DESCRIPTION

The M5M5165P is a 65,536-bit CMOS static RAM organized as 8,192 words by 8 bits which is fabricated using high-performance double polysilicon CMOS technology. The use of resistive load NMOS cells and CMOS periferals result in a high-density and low-power static RAM. It is ideal for the memory systems which reguire simple interface.

The stand-by current is low enough for a battery back-up application. It is mounted in a standard 28 pin package and configured in an industrial standard 8K x 8-bit pinout.

FEATURES

Type	Access time (max)	Power supply current	
		Active (max)	Stand-by (max)
M5M5165P-70	70ns	50mA	2mA
M5M5165P-10	100ns		
M5M5165P-12	120ns		
M5M5165P-70L	70ns		100μA
M5M5165P-10L	100ns		
M5M5165P-12L	120ns		

- Single +5V Power Supply
- Fully Static Operation: No Clocks, No Refresh
- Data-Hold on +2V Power Supply
- Directly TTL Compatible: All Inputs and Outputs
- Three-State Outputs: OR-tie Capability
- Simple Memory Expantion by $\overline{S_1}$, S_2
- $\overline{OE}$ Prevents Data Contention in The I/O Bus
- Common Data I/O
- Pinout Compatible with 64K EPROM M5L2764K

APPLICATION

Small Capacity Memory Units.

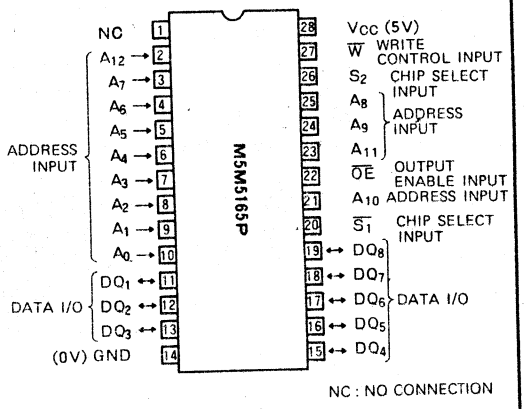
FUNCTION

The operation mode of the M5M5165P is determined by a combination of the device control inputs $\overline{S_1}$, S_2, $\overline{W}$ and $\overline{OE}$. Each mode is summarized in the function table. (see next page)

A write cycle is excuted whenever the low level $\overline{W}$ overlaps with the low level $\overline{S_1}$ and the high level S_2. The address must be set up before the write cycle and must be stable during the entire cycle. The data is latched into a cell on the trailing edge of $\overline{W}$, $\overline{S_1}$ or S_2, whichever occurs first, requring the set-up and hold time relative to these edge to be maintained. The Output enable input $\overline{OE}$ directly controls the output stage. Setting the $\overline{OE}$ at a high level, the output stage is in a high-impedance state, and the data bus contention problem in the write cycle is eliminated.

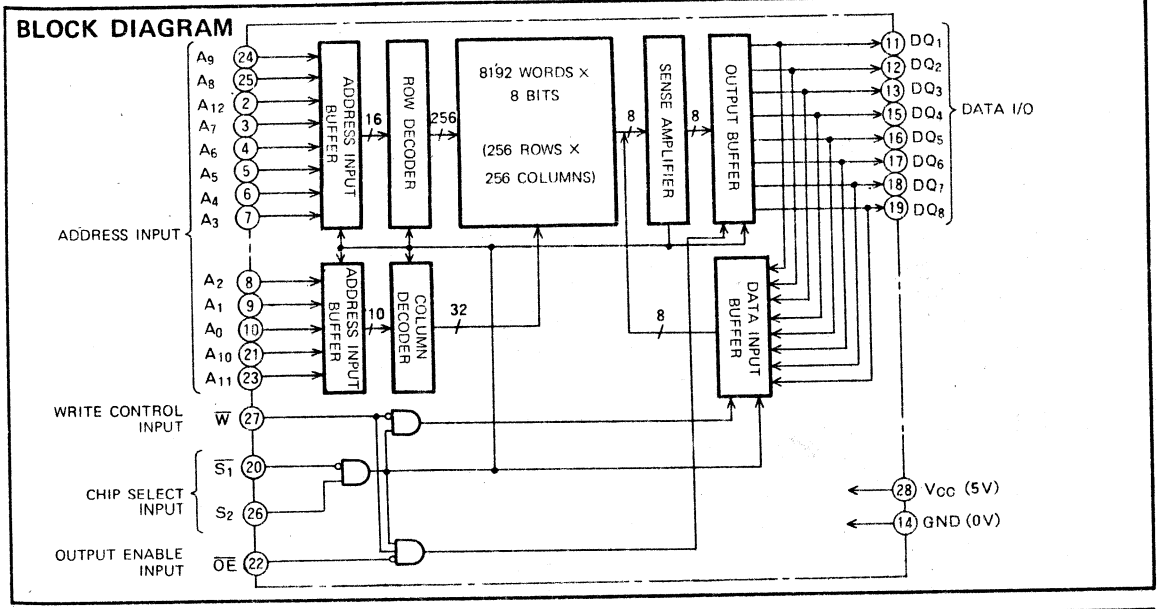
A read cycle is excuted by setting $\overline{W}$ at a high level and $\overline{OE}$ at a low level while $\overline{S_1}$ and S_2 are in an active state ($\overline{S_1}$ = L,

PIN CONFIGURATION (TOP VIEW)



Outline 28P4

BLOCK DIAGRAM



S_2=H)

When setting $\overline{S_1}$ at a high level or S_2 at a low level, the chip is in a non-selectable mode in which both reading and writing are disabled. In this mode, the output stage is in a high-impedance state, allowing OR-tie with other chips and memory expantion by $\overline{S_1}$ and S_2. The power supply current is reduced as low as the stand-by current which is specified as I_{CC3} or I_{CC4}, and the memory data can be held at +2V power supply, enabling battery back-up operation during power failure or power-down operation in the non-selected mode.

FUNCTION TABLE

$\overline{S_1}$	S_2	$\overline{W}$	$\overline{OE}$	Mode	DQ	I_{CC}
X	L	X	X	Non selection	high-impedance	Standby
H	X	X	X	Non selection	high-impedance	Standby
L	H	L	X	Write	D_{IN}	Active
L	H	H	L	Read	D_{OUT}	Active
L	H	H	H		high-impedance	Active

RECOMMENDED OPERATING CONDITIONS (T_a=0～70°C, unless otherwise noted)

Symbol	Parameter	Limits			Unit
		Min	Typ	Max	
V_{CC}	Supply voltage	4.5	5	5.5	V
GND	Supply voltage		0		V
V_{IL}	low input voltage	−0.3		0.8	V
V_{IH}	high input voltage	2.2		V_{CC}+0.3	V

ABSOLUTE MAXIMUM RATINGS

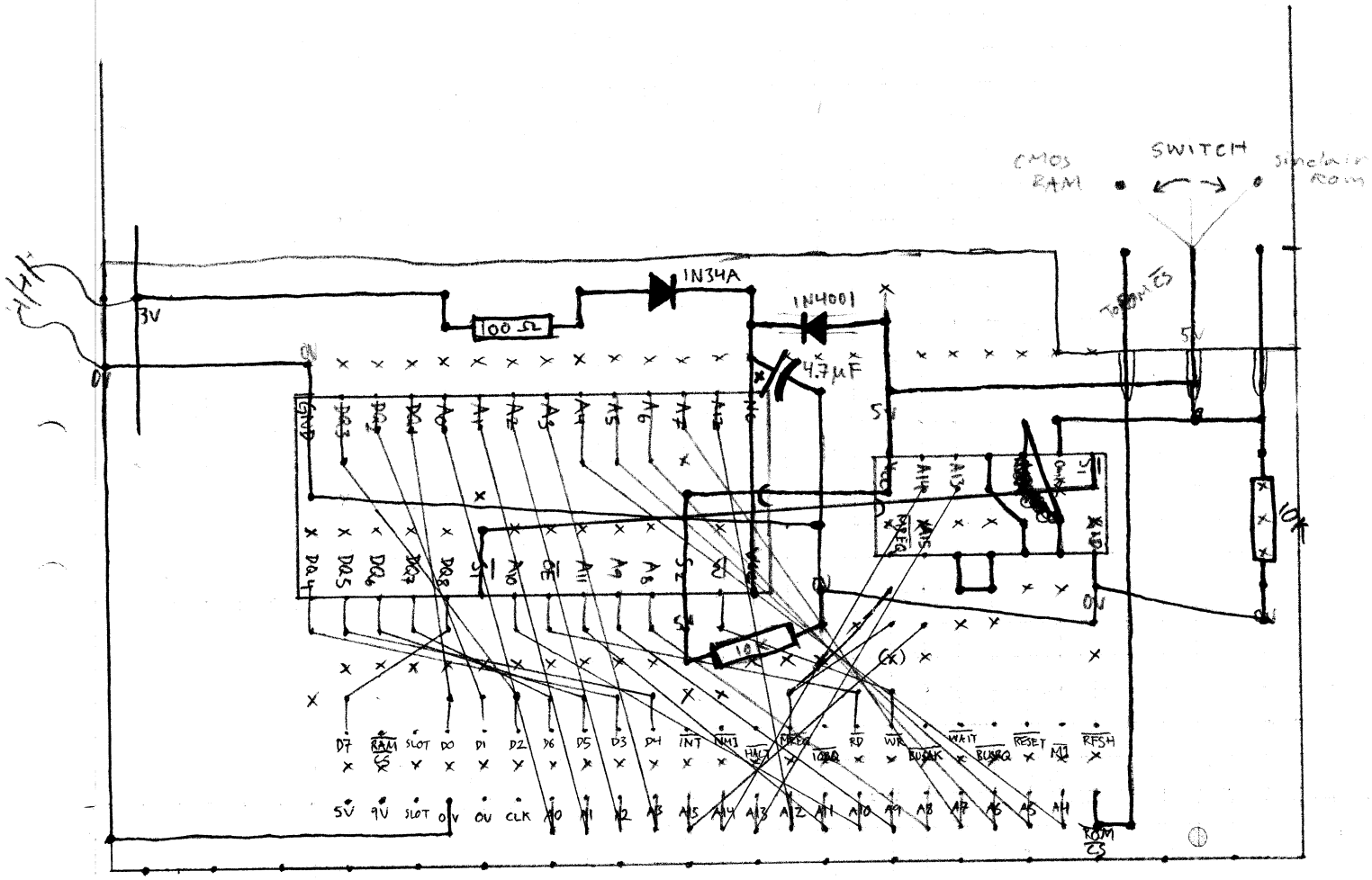
Symbol	Parameter	Conditions	Limits	Unit
V_{CC}	Supply voltage	With respect to GND	−0.3～7	V
V_I	Input voltage	With respect to GND	−0.3～V_{CC}+0.3	V
V_O	Output voltage	With respect to GND	0～V_{CC}	V
P_d	Power dissipation	T_a=25°C	700	mW
Topr	Operating temperature		0～70	°C
Tstg	Storage temperature		−65～150	°C

ELECTRICAL CHARACTERISTICS (T_a=0～70°C, V_{CC}=5V±10%, unless otherwise noted)

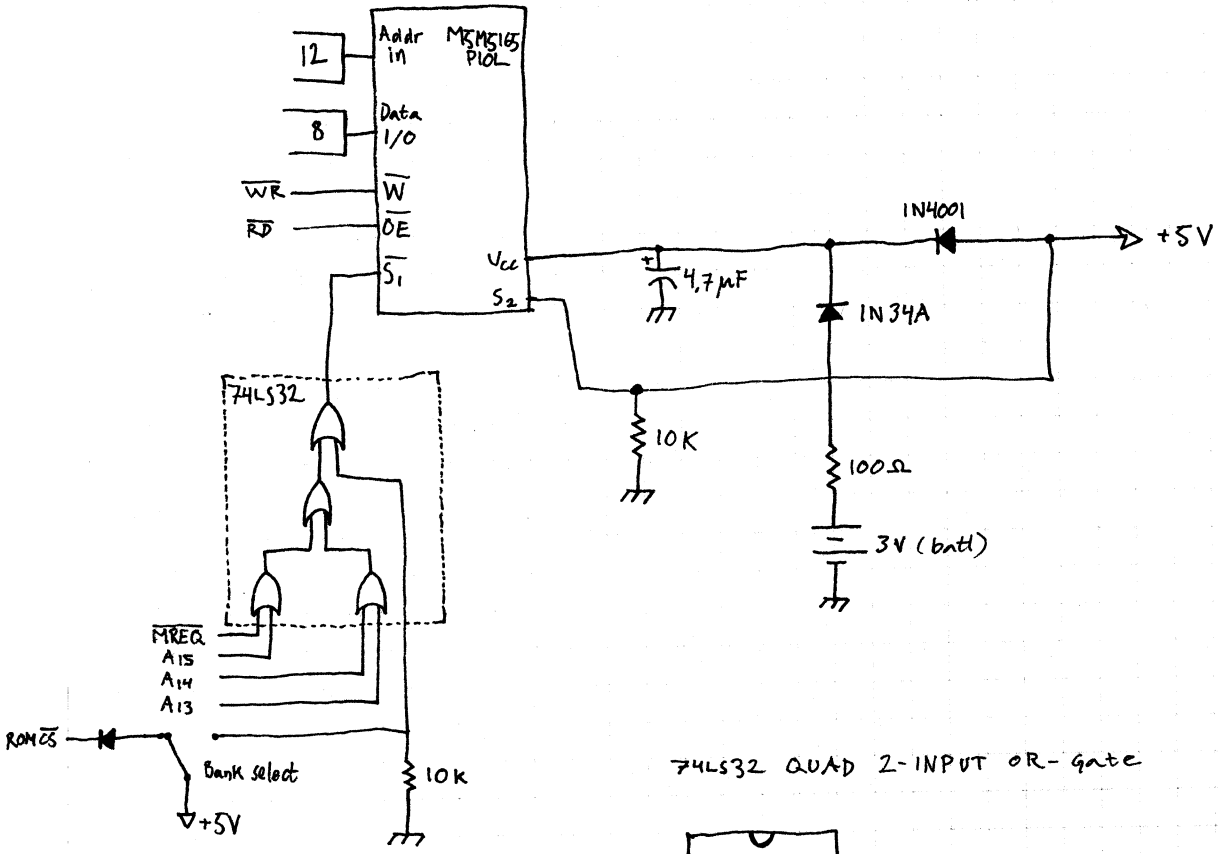
Symbol	Parameter	Test conditions	Limits			Unit
			Min	Typ	Max	
V_{IH}	High input voltage		2.2		V_{CC}+0.3	V
V_{IL}	Low input voltage		−0.3		0.8	V
V_{OH}	High output voltage	I_{OH}=−1mA	2.4			V
V_{OL}	Low output voltage	I_{OL}=2mA			0.4	V
I_I	Input current	V_I=0～V_{CC}			±1	μA
I_{OZH}	High level output current in off-state	$\overline{S_1}$=V_{IH} or S_2=V_{IL} or $\overline{OE}$=V_{IH}, $V_{I/O}$=0～V_{CC}			1	μA
I_{OZL}	Low level output current in off-state	$\overline{S_1}$=V_{IH} or S_2=V_{IL} or $\overline{OE}$=V_{IH}, $V_{I/O}$=0～V_{CC}			−1	μA
I_{CC1}	Active supply current	$\overline{S_1}$≦0.2, S_2≧V_{CC}−0.2, Outout open, Other inputs ≦0.2 or ≧V_{CC}−0.2		30	45	mA
I_{CC2}	Active supply current	$\overline{S_1}$=V_{IL} or S_2=V_{IH}, Output open, Other inputs =V_{IH}		35	50	mA
I_{CC3}	Stand-by supply current	① S_2≦0.2V, Other inputs=0～V_{CC} ② $\overline{S_1}$≧V_{CC}−0.2V, S_2≧V_{CC}−0.2V, Other inputs =0～V_{CC}			2(P)	mA
I_{CC4}	Stand-by supply current				100(P-L)	μA
C_I	Output capacitance (T_a=25°C)	S_2=V_{IL}, $\overline{S_1}$=V_{IH}, Other inputs=0～V_{CC}			3	mA
C_O	Outout capacitance (T_a=25°C)	V_I=GND, V_i=25mVmrs, f=1MHz			6	pF
		V_O=GND, V_O=25mVrms, f=1MHz			8	pF

Note 1 Direction for current flowing into IC is indicated as positive (no mark)
2 Typical value is V_{CC}=5V, Ta=25°C

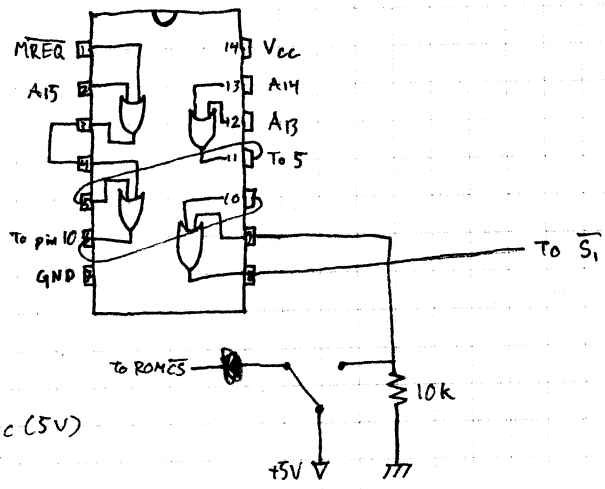
Kretskortslayout för Batteribackup CMOS Rana

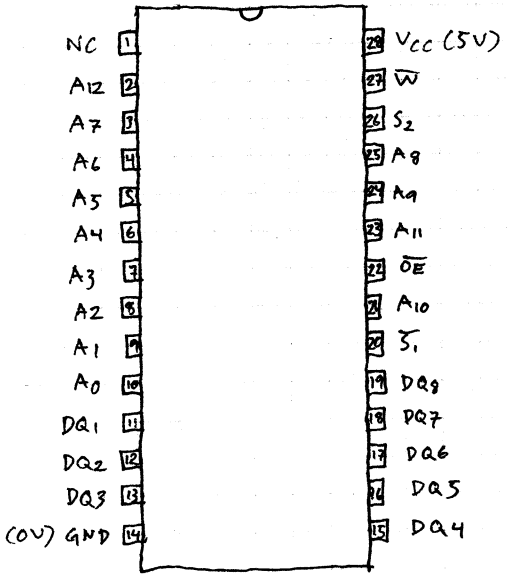


Battery backup CMOS RAM



74LS32 QUAD 2-INPUT OR-Gate





TEMPERATURE-ANALOGUE INTERFACE

BY R. A. PENFOLD

Two-channel temperature interface plus dual analogue ports for the Amstrad and the Spectrum

THIS interface was designed to give the Amstrad CPC464 computer four analogue inputs, two of which are intended for use with semiconductor temperature sensors to give a measuring range of 0 to 51 degrees Centigrade with 0·2 degree resolution. However, the unit can be used as a straightforward four channel analogue to digital converter if desired.

Although the unit was not designed with the Spectrum computer in mind, it has been tried with this computer and worked perfectly well. The only modification required for use with the Spectrum is the use of a different edge connector.

The two analogue inputs have a full scale sensitivity of 1·2V and can be used together to provide a single differential input. If the inputs for the two temperature sensors are used as ordinary analogue inputs they have a full scale value which is adjustable from about 350mV to 1·2V. The sensitivity of each channel is individually adjustable. All four channels offer an accuracy of plus and minus 1 l.s.b.

SYSTEM OPERATION

A circuit of this type could be quite complex, but in this case things are kept remarkably simple by using a modern four-channel analogue to digital converter chip (the ADC0844CCN) and an equally modern temperature sensor chip (the LM35DZ). Fig. 1 shows the block diagram for the interface, and several of these blocks represent the internal circuit of the ADC0844CCN.

At the heart of the unit is the single successive approximation converter of the ADC0844CCN. The way in which this type of converter functions has been described in previous articles in this magazine, and we will not consider this aspect of the unit in detail here. While not providing ultra-fast conversions, the successive approximation technique is reasonably fast and can be achieved by a relatively low cost device. In the case of the ADC0844CCN the conversion time is no more than 40µs, and at 25 degrees Centigrade is typically 30µs. This enables up to about 25000 conversions per second to be achieved, which is more than adequate for the vast majority of applications. The reason for the exact conversion time being temperature dependent is that the ADC0844CCN has a built in clock oscillator which includes on chip timing components. The lack of frequency stability that this inevitably produces is of little practical importance, and it helps to minimise the number of discrete components that are needed.

A four-channel analogue multiplexer ahead of the converter provides the unit with its four input channels. This is not as good as having four separate converters as each channel in use has to be read in turn. With all four channels in use this enables each channel to be read only about 6000 times or so per second, but even this reduced rate is far higher than is required for most applications.

The required channel is selected by writing data to the converter, but only the four least significant bits are utilized when doing this. The data is stored in a four-bit latch, and in addition to selecting the channel to be converted it also sets the operating mode of the device and initiates a conversion. There are three operating modes, and these are single ended, differential, and pseudo differential. The single ended mode is the normal one where each input responds to the input voltage relative to the earth rail. In the differential mode the converter becomes a two-channel type with one pair of input voltages applied to chan-

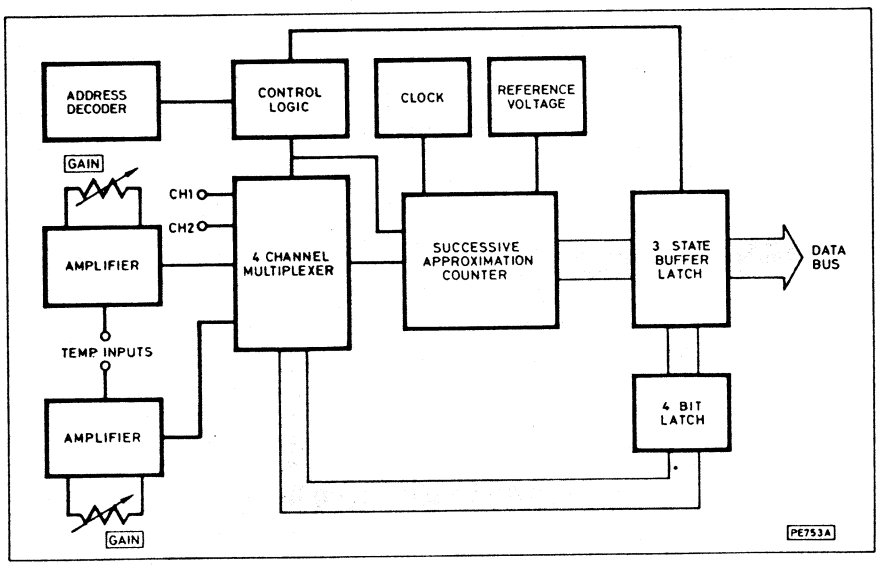


Fig. 1. *Block diagram*

Unlike many analogue to digital converter chips, the ADC0844CCN does not include an internal reference voltage source. The full scale value of the converter is equal to the reference voltage used, and this can be any voltage of 5V or less. In the interest of good linearity it is not advisable to use a reference potential of much less than about 1V though. In this circuit a highly stable 1·2V reference source is utilized.

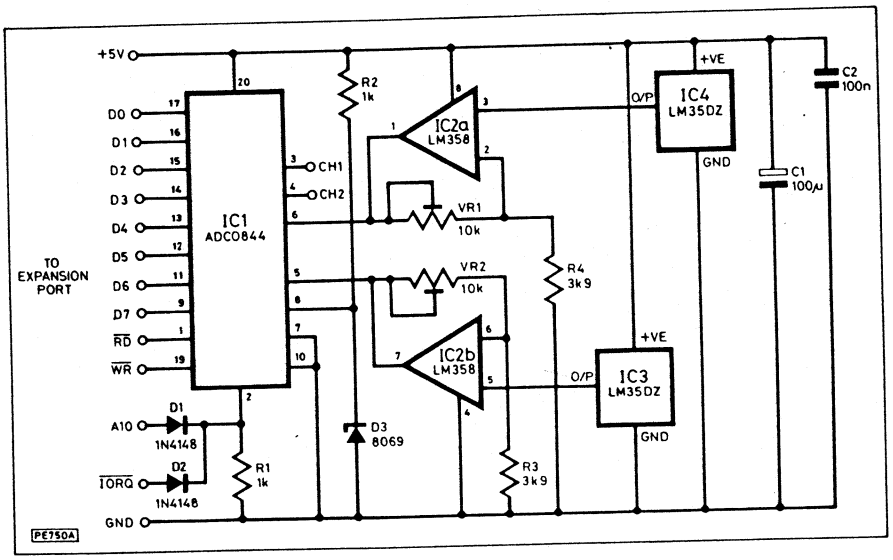


Fig. 2. *Complete circuit diagram*

nels 1 and 2, and the other two applied to channels 3 and 4. In the pseudo differential mode the unit is reduced to three channels (1, 2, and 3) with channel 4 being used as the inverting input for all three channels. This mode is not really applicable in this case due to the amplifiers added ahead of the channel 3 and channel 4 inputs.

When a conversion has been completed the result is stored in an eight-bit data latch which has three state outputs. The outputs of the latch can therefore be connected direct to the data bus of the computer. Although the converter has four channels it only occupies one address, and the channel read at that address is the one previously selected by a write operation. Even if the same channel is read each time, a write operation must precede the taking of each reading in order to initiate fresh conversions. Otherwise the same conversion would be read repeatedly.

The address decoder is extremely simple, and only decodes one address line plus one control line. This is made possible by the simple system of input/output mapping used in the CPC464 and Spectrum computers, plus the fact that the ADC0844CCN is specifically intended to interface with 8080/Z80 based systems, and has inputs for the $\overline{RD}$ (read) and $\overline{WR}$ (write) lines.

TEMPERATURE SENSOR

There are normally difficulties if a semiconductor temperature sensor is connected direct to the input of an analogue to digital converter without using some form of signal conditioning circuit. For example, the popular LM335Z provides an output of 10mV per degree Kelvin, or 2·73V plus 10mV per degree Centigrade in other words. A temperature range of (say) 0 to 50 degrees Centigrade therefore gives an output voltage range of 2·73V to 3·23V. Obviously the output of the sensor could be fed to an analogue input having a full scale value of 3·23V, but this would be a very inefficient way of doing things since input voltages from 0V to 2·73V would never occur, and most of the converter's input range would be wasted. This is not merely of academic importance, and in practice would result in the system having relatively poor resolution and accuracy.

Much better results could be obtained using a signal conditioning circuit to remove the 2·73V offset, and to spread out the 0 to 50 degree range over the full input voltage range of the converter. This type of manipulation is rendered unnecessary by the LM35DZ temperature sensors used in this design. These have a built-in circuit to eliminate the offset voltage, so that the output voltage is nominally 10mV per degree Centigrade. This gives a 0 to 510mV output voltage range over the 0 to 51 degrees Centigrade temperature span covered in this case. A simple d.c. amplifier at each of the two temperature sensor inputs boosts this to a 0 to 1·2V range which fully drives the converter. The converter provides readings of 0 to 255, which conveniently converts to readings in degrees Centigrade simply by dividing returned values by five. The gain of both amplifiers has been made variable so that they can be set up for optimum accuracy.

It must be pointed out that at very low temperatures of only about 1 degree Centigrade the LM35DZ may not give good accuracy, but it will give good results over a range of 2 to 51 degrees Centigrade. In its favour it has the advantage of not requiring a highly stable supply, or even a stabilised supply for that matter, and it will operate over a 4 to 30 volt supply range. With a current consumption of only 56µA there is little self heating.

CIRCUIT OPERATION

The full circuit diagram of the interface appears in Fig. 2. D3 and R2 form a shunt regulator which provides the 1·2V reference for converter chip IC1. D3 is not an ordinary Zener diode, but is a highly accurate and temperature compensated regulator chip. IC2 provides the two amplifiers which are conventional operational amplifier non-inverting mode types. The LM358 is one of the few operational amplifiers that is capable of providing output voltages right down to the earth rail, and this renders a negative supply rail unnecessary. IC3 and IC4 are the two LM35DZ temperature sensors.

The unit connects to the floppy disc port of the CPC464 computer, which is really a general purpose expansion port having the full buses etc. available. The system of input/output mapping used in the CPC464 is to have circuits activated by taking one the eight most significant address lines low. One or more of the eight least significant address lines can be decoded as well if an input/output circuit requires several addresses (which is obviously not the case here). Address line A10 is available for external add-ons, and this line is decoded with the $\overline{IORQ}$ (input/output request) line and used to generate the negative chip select pulse for IC1. D1, D2, and R1 form a simple 2-input OR gate. The interface is placed at address &F800 in the

COMPONENTS . . .

RESISTORS

R1,R2	1k (2 off)
R3,R4	3k9 (2 off)
All ¼W 5% carbon	

POTENTIOMETERS

VR1,VR2	10k 0·1W horizontal preset (2 off)

CAPACITORS

C1	100µ 10V radial elect
C2	100n ceramic

SEMICONDUCTORS

D1,D2	1N4148 (2 off)
D3	8069
IC1	ADC0844CCN
IC2	LM358
IC3,IC4	LM35DZ (2 off)

MISCELLANEOUS

Case about 133 by 70 by 38mm; printed circuit board, PE 101; computer connector; 20 pin d.i.l. i.c. holder; 8-pin d.i.l. i.c. holder; stereo jack socket (3 off); stereo jack plug (3 off); 14-way ribbon cable; connecting wire; solder, etc.

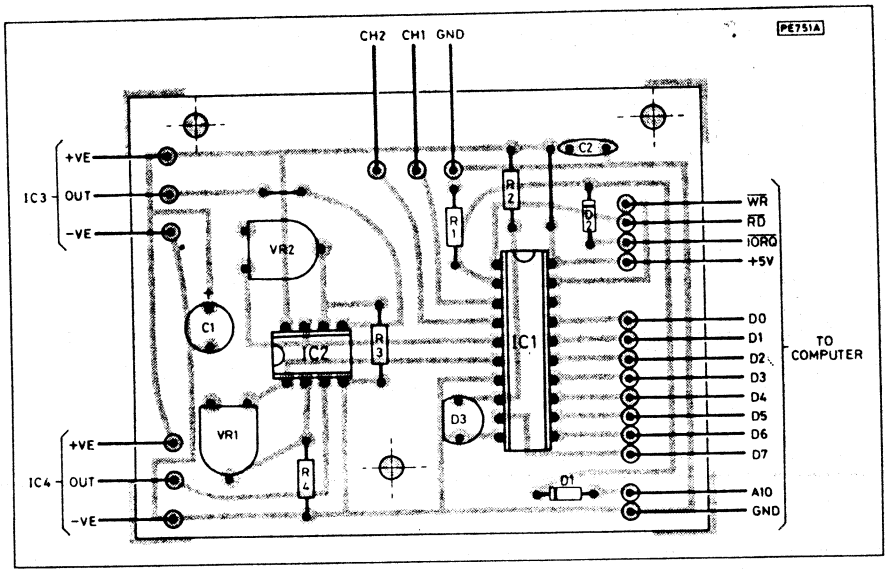


Fig. 3. *P.c.b. design and component layout*

input/output map. Of course, as there is only partial address decoding the circuit can be activated using a large block of addresses, but the one mentioned above is the base address and is a convenient one to use in practice.

In essence the system of input/output mapping adopted for the Spectrum is the same as that used in the CPC464. There is a slight difference in that one of the lower eight address lines is taken low to activate input/output circuits, and one or more of the upper eight address lines are used if additional addresses are required. For the Spectrum it is address line A5 that is gated with the $\overline{\text{IORQ}}$ line, and address 65503 that the circuit occupies. Again, due to the partial address decoding the circuit occupies a great many addresses, but in practice 65503 is probably the best one to use.

POWER SUPPLY

The circuit requires a single (+5V) supply. The current consumption is only a few milliamps, and both the CPC464 and the Spectrum can readily supply this.

CONSTRUCTION

Fig. 3 shows the printed circuit board design for this interface. An important point to note is that IC1 is a CMOS device and should accordingly be fitted in a (20 pin d.i.l.) i.c. holder. The other standard antistatic handling precautions should also be observed. There are two link wires which should not be overlooked, but in other respects construction of the board is quite ordinary.

The exact form that mechanical construction of the unit takes will depend on how it is to be used. It can simply be left as a loose board or, like the prototype, it can be built into a case. A third alternative is to incorporate the unit in to some larger piece of equipment. Whatever method is used the printed circuit design of Fig. 3 should be satisfactory.

Assuming that the unit is to be built into its own case, an aluminium box having approximate outside dimensions of 133 by 70 by 38mm is suitable. The temperature sensors would not normally be mounted on the printed circuit board, but would be remotely located and connected to the interface by way of three-way cable. In order to minimise problems with stray pick up of noise it is advisable to use twin overall screened cable, with the screen carrying the negative supply rail connection. On the prototype three stereo jack sockets are mounted on the front panel, and connections from the sensors to the printed circuit board are made via two of these. The third one is used for the two ordinary analogue inputs.

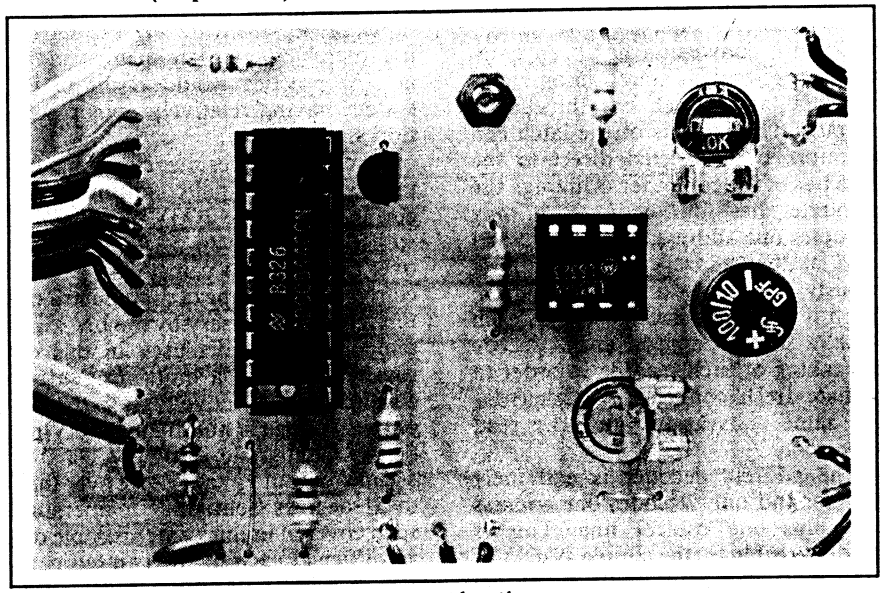


Photo 1. *P.c.b. and component mounting details*

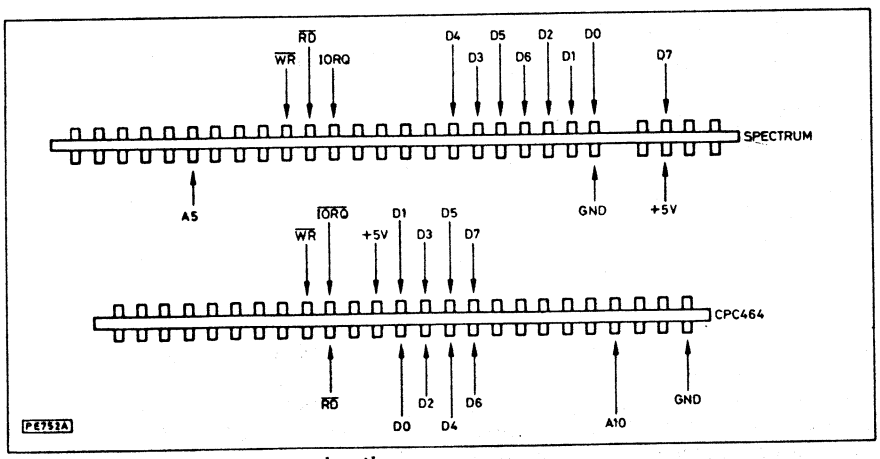


Fig. 4. *Computer connection details*

A 14-way ribbon cable up to about one metre long connects the printed circuit board to the floppy disc port of the CPC464 or the expansion port of the Spectrum. For the Amstrad computer a 2 by 25-way 0·1 inch pitch edge connector is required. As it is unlikely that a connector having a suitable polarising key will be available care must be taken to fit the connector the right way up, and it is advisable to clearly mark the top edge. The Spec-

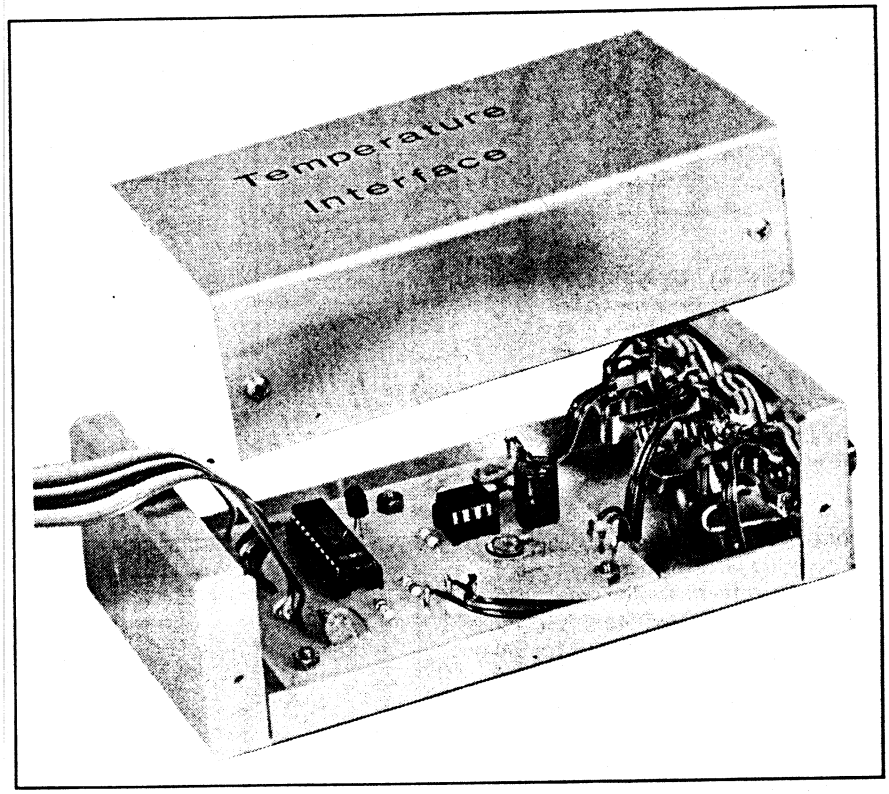


Photo 2. *Constructional details of the Temperature Interface*

trum's expansion port requires a 2 by 28-way 0·1 inch pitch edge connector, and types having a suitable polarising key are readily available. Fig. 4 gives connection details for both computers.

Take great care not to make any wiring errors, and thoroughly check all the wiring once it has been completed. With the specified case it is possible to take the ribbon cable out between the top and base sections, but if an alternative is used it might be necessary to cut an exit slot in the rear panel.

TESTING AND USE

Start with VR1 and VR2 set at about half maximum resistance. Connect the interface to the computer before switching on the computer. The two simple programs shown in Table 1 can be used to give a temperature readout in degrees Centigrade for both temperature channels.

From BASIC, the OUT instruction is used to write the appropriate number to the converter to start a conversion and select the required channel. The number is 6 to read IC4 and 7 to read IC3. The INP function (IN for the Spectrum) is used to read the converter, and as explained earlier, dividing the reading by five gives a reading in degrees Centigrade. The relative slowness of BASIC means that a conversion will always have been completed by the time the unit is read, but when using machine code a delay loop to provide a hold off of at least 40μs will be required.

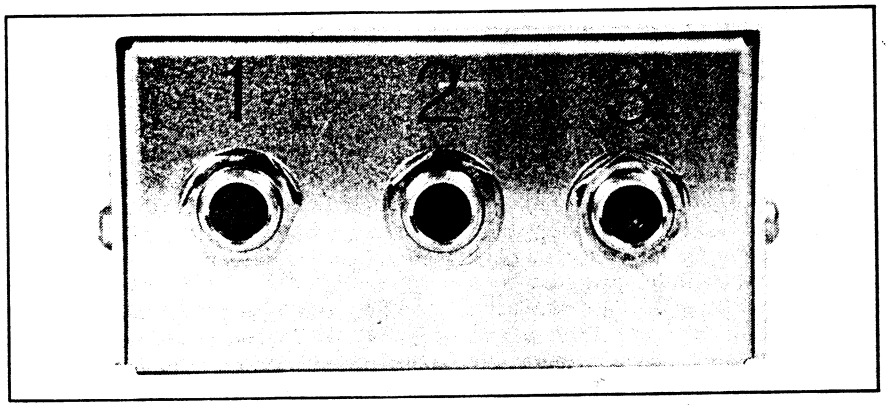
In order to calibrate the unit both sensors should be taken to a known temperature, and one that represents about 50% to 100% of the full scale value. The two presets are then adjusted to give the correct reading. If the sensors are to be used in liquids they should be fitted in a small test tube or a similar container to protect them. Some silicon grease can be used to give a good thermal contact between each sensor and its container.

In the single ended mode analogue channels 1 and 2 are selected using values of 4 and 5 respectively. When these are used as a single differential input they are selected using values of 0 (channel 2 is the inverting input) or 1 (channel 1 is the inverting input). PE

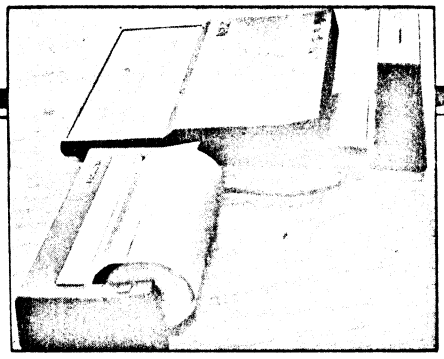
Table 1. *Amstrad listing*

```
10 REM CPC464
20 MODE 0
30 LOCATE 8,5
40 PRINT "DEGREES C."
50 OUT &F800,6
60 LOCATE 5,10
70 PRINT (INP(&F800))/5
80 OUT &F800,7
90 LOCATE 15,10
100 PRINT (INP(&F800))/5
110 FOR D = 1 TO 1000:NEXT
120 LOCATE 5,10
130 PRINT "      "
140 LOCATE 15,10
150 PRINT "      "
160 GOTO 50


10 REM SPECTRUM
20 PRINT AT 8,10;"DEGREES C."
30 OUT 65503,6
40 PRINT AT 16,7;IN 65503/5
50 OUT 65503,7
60 PRINT AT 16,20;IN 65503/5
70 PAUSE 50
80 PRINT AT 16,7;"      "
90 PRINT AT 16,20;"      "
100 GO TO 30
```

Photo 3. *Front panel details*

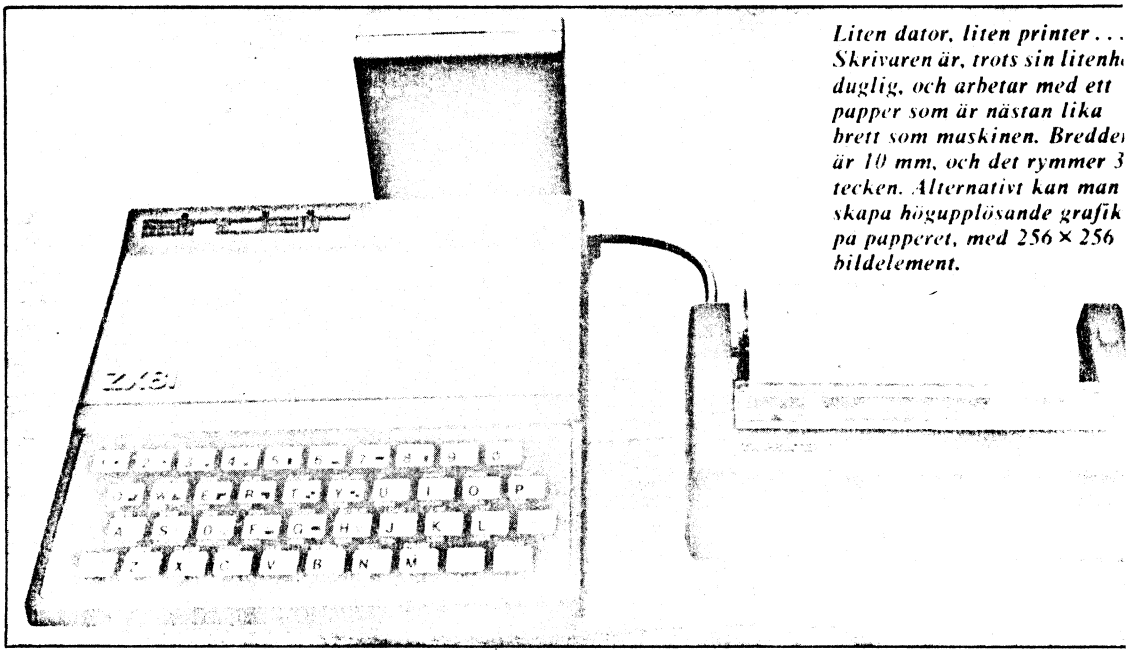


ZX Printer från Sinclair



Skrivaren ansluts datorn baktill på bussen. Anslutningsdonet är så utformat att man samtidigt kan sluta t ex ett yttre minn. Det stora minnet är en förutsättning för den högupplösande grafiken.

Med skrivare kan man dokumentera program och programkörningar. Den skrivare som nu kommer till ZX81 kan emellertid mer än så. Man kan också få ut hela skärmbilder, inklusive grafik, på papperet. Med ett särskilt styrprogram i delvis maskinkod kan man också få ut verkligt högupplösande grafik!



Liten dator, liten printer... Skrivaren är, trots sin litenhe duglig, och arbetar med ett papper som är nästan lika brett som maskinen. Bredde är 10 mm, och det rymmer 3 tecken. Alternativt kan man skapa högupplösande grafik på papperet, med 256 × 256 bildelement.

■■ Att *ZX81* blivit en mycket populär dator bör ingen förvåna sig över. Trots att datorn har ett synnerligen enkelt tangentbord och väldigt litet minne i grundutförande så kan den ändå i stort sett lika mycket som större maskiner. Och det till ett, hittills, oslagbart lågt pris. Det priset höjs med åtskilliga hundralappar när man finner att det inbyggda arbetsminnet inte räcker till för några ståtligare programövningar.

Trots det kan man tänkas ha pengar över att köpa en skrivare till datorn också. En konventionell skrivare kostar från 2 000–3 000 kronor och uppåt. Sinclairs kostar ca 1 000 kr. Då är inte utseendet eller funktionen densamma som hos en mera konventionell skrivare, den har ändå en del egna kvaliteter.

Till att börja med är den, på Sinclair-vis, mycket liten. Datorn är ju heller ingen utrymmesbjässe, och skrivaren passar väl in i systemet. Den största delen av skrivaren är faktiskt nästan pappersrullen.

Om man skall göra en billig skrivare vill man helst slippa den konventionella skrivmetoden med typer och färgband. De kostar en hel del pengar. I stället kan man välja endera av termofunktion, eller skrivare med metalliserat papper. Sinclair har valt det metalliserade papperet.

Bäst resultat får man alltid med färgband och tryckverk. Den skriften blir tålig och beständig, om man valt ett färgbandsbläck som är beständigt. Man skriver också på vanligt papper som är billigt och ger en lättläst skrift med hög kontrast.

På termopapperet skriver man genom att hetta upp de punkter man vill ha mörka. Skrivhuvudet består av en punktmatris vars punkter hettas upp elektriskt. Här krävs ett specialpreparerat papper som mörknar vid upphettning. Skriften blir av god kontrast, men papperet kan mörkna om det värms upp oavsiktligt. Det kan räcka med värme från en stark lampa eller värmen ovanpå en elektrisk apparat. Skriften bleks också gärna i starkt ljus.

Det metalliserade papperet har svart framsida. Ovanpå det svarta har man anbringat ett tunt metallskikt, som är elektriskt ledande. Metallskiktet är ljust, och täcker papperet. Genom att leda en tillräckligt stark ström genom metallskikten kan man bränna bort det så att svärtan under blir synlig. Så kan man bygga upp svart text på ljus botten. Kontrasten kan bli dålig i vissa lägen men är för det mesta alldeles tillräcklig. Hållbarheten är däremot god. Texten bleks inte av vare sig höga temperaturer eller solljus. Däremot kan ytan skadas genom skrapning så man får en massa extra streck. Det går bra att skriva med penna på papperet. Man kan t o m skriva med en pinne och repa hål i det tunna metallskiktet!

För att det skall bli lätt att läsa den skrivna texten krävs att raderna inte är för korta. Smalt papper och korta rader är annars ett sätt att göra skrivaren billig. Det papper Sinclair använder är ingalunda av standardiserad A4-bredd, men har ändå rätt rejält format. Bredden är 100 mm, och på den kan man skriva 32 tecken. Det är precis lika mycket som på skärmen, och papperstext och skärmtext blir lika läsbara. De blir faktiskt också helt lika eftersom både papper och skärm visar svarta tecken på vit botten. De flesta andra bildskärmar visar vita tecken på svart botten.

Det finns två enkla sätt att få något skrivet på papperet. Det ena är att man använder kommandot **LPRINT** eller **LLIST.** Med **LPRINT** i ett program kan man göra precis samma utskrifter på papper som man annars kan på skärmen. Med **LLIST** kan man få en programlista på papper. Man kan styra vid vilken rad listningen skall börja, genom att skriva **LLIST, radnummer.** Så kan man också göra med vanliga **LIST** på skärmen. Där stoppas emellertid listningen upp när skärmen är full. På skrivaren slutar listningen först när programmet är slut.

Den andra, mera ovanliga möjligheten med ZX81 och dess skrivare är att man med ett enda kommando kan överföra skärmen precis som den är, till papperet. Man kan alltså överföra inte bara text, utan alla de grafiska krumbukter man tillåtit sig också. Kommandot heter **COPY**, och innebär alltså kort och gott att skärmbilden överförs på papper.

För grafik har papperet alltså samma upplösning som skärmen,

forts på sid 88

radio &
television
Box 3224
103 64 Stockholm 3

radio &
television
Box 3263
103 65 STOCKHOLM

Brev-
porto

Informationstjänsten
radio &
television
Box 3224
103 64 Stockholm 3

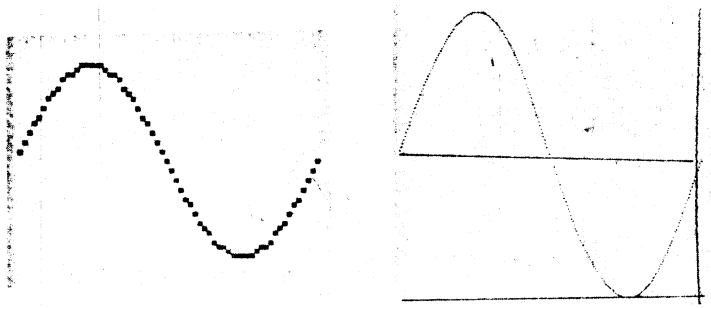
ZX-PRINTERN
forts fr sid 5

dvs 64 gånger 44 fritt adresserbara punkter eller 2 816 bildelement. Genom diverse tricksande med maskinkoder kan man emellertid bygga upp bilder med 256×256 punkters upplösning på papperet! Det är en synnerligen god, högupplösande grafik, helt i klass med vad **Apple** och **ABC 800** t ex presterar. Grafiken är förstås inte i färg, och den blir väldigt långsam, men möjligheten finns där och är mycket spännande att jobba med. För att skriva den högupplösande grafiken måste man ha stort minne. 256×256 punkter motsvarar 65 536 bildpunkter, som var och en måste lagras i en bit. Det går alltså åt 8 Kbyte bara för bildlagringen, och till det kommer program. Skrivaren i sig kräver annars inget extra minne, utan kan också användas på maskiner med bara 1 K minne. Däremot får skrivaren sin strömförsörjning från datorn, och därmed från nätaggregatet. Det ursprungliga aggregatet räcker dock inte till, varför ett starkare sådant följer med skrivaren.

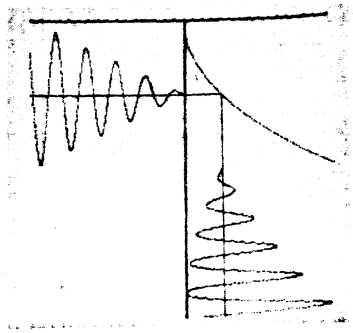
Den kanske viktigaste användningen för skrivaren är att man kan få ut programmen på en lista. När man gjort ett långt program är det mödosamt att bläddra i de 22 raderna på skärmen för att se vad man gjort för fel. Med hela programmet på en lista har man en betydligt bättre överblick. Man kan då i lugn och ro pricka av alla förändringar man vill göra, och riskerar mindre att glömma bort någon. Det kan också vara fördelaktigt att spara programlistan på papper, även om man sparar själva programmet på band.

Utskrifter från programkörningar är också bra att ha i många fall. Om man t ex gör ett budgetprogram för sin privata ekonomi är det närmast nödvändigt att kunna dokumentera körningarna på något sätt. Det kan man naturligtvis göra genom att skriva av skärmen, men allra enklast är förstås att låta skrivaren göra det automatiskt.

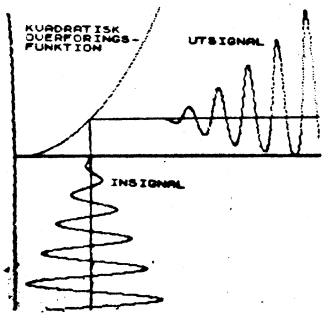
ZX81 växer alltså till sig. Nu kan man ansluta både ett rejält minne och en bra skrivare. Vi väntar bara på en flexskiva för under tusenlappen också. Men någon sådan verkar inte vara i faggorna, så vi får väl roa oss med skrivaren så länge. Och det går ju alldeles utmärkt, som synes av våra utskrifter! **BH**



Här kan vi jämföra utskrift av grafik på skrivaren, med två metoder. Den grövre bilden är en direkt utskrift av en bild som skapats med PLOT-funktionen på skärmen. Den finare är gjord med programmet för högupplösande grafik.



Här är ett annat exempel på grafiken. Bilden föreställer funktionen hos en förstärkare med kvadratisk karakteristik. Insignalen kommer nederst, "speglas" av den kvadratiska överföringskurvan, och går ut till höger. En sådan här bild skulle man gärna vilja förse med text också. Tyvärr går det inte att blanda text med bilden i högupplösningsgrafiken, annat än att man definierar varje tecken för sig.



Här har vi blandat text och grafik, trots att det inte "går". Knepet är att vi överlagrat två bilder på papperet. Först skrev vi ut kurvan, sedan drog vi tillbaka papperet till utgångsläget, och skrev texten med en ny programsnutt. Det gäller bara att passa in texten så att den passar! Därför bör man knappast laborera med text som skall passa så där väldigt exakt, men om man drar tillbaka papperet så att dess överkant ligger jäms med rivränderna så blir noggrannheten förvånande god.

```
1 REM LPRND.LPRNDLP .TAN ..
```

Adress	Dec	hex	Beskrivning	
16514	58	3A	LDA, 408E	innehållet i 408E (=16 526) till processorns
16515	142	8E		A-register
16516	64	40		
16517	95	5F	LDE, A	innehållet i processorns A-register till dess E-register
16518	58	3A	LDA, 408F	innehållet i 408F (=16 527) till A-registret
16519	143	8F		
16520	64	40		
16521	179	B3	OR E	jämför innehållen i A och E, resultat i A
16522	6	06	LDB, 00	ladda B med 00, dvs nollställ B
16523	0	00		
16524	79	4F	LDC, A	innehållet i register A till C
16525	201	C9	RET	återvänd från subrutinen

Det här är maskinkodrutinen i rad 1 på grafikprogrammet. Raden måste skrivas in just som den är visad här. Men hur kan man skriva ett maskinkodprogram i en REM-sats?

Ett maskinkodprogram måste ligga på ett bestämt avsnitt i minnet, annars stämmer inte anropsadresserna bl a. För att kunna lägga det mitt i ett basicprogram så att man kan lagra på band t ex, måste man lägga det på ett ställe där det inte flyttar på sig. Om man redigerar i ett program kan satserna flytta sig i minnet, utan att man egentligen tänker på det. Första raden ligger däremot still. Och man vet vilken adress den alltid kommer på.

Sedan radnummer och REM-kommando är avräknat, kommer det första tecknet i satsen att hamna på adressen 16 514, och resten framåt.

Men vad betyder då de underliga tecknen? Ja, de betyder i sig ingenting annat än siffror. Ett sätt att slå in en sifferkod i datorn är att slå in ett kommando. Kommandot RND motsvarar t ex talet 64. Raden av konstiga tecken är alltså en rad siffror, som i sin tur motsvarar maskinspråkinstruktionen.

På ett par ställen i rad 1 står här en punkt. De ställena omvandlas vid körningen till andra tecken, med POKE-satser i programmet. Men varför då inte slå in de tecken man vill ha från början? Svaret är att det inte går! För de koderna finns ingen funktion! När man kört programmet och sedan listar det får man visserligen ut tecken i de positionerna, men tecknen är inte entydigt bestämda, utan kan motsvaras av flera koder. Därför kan man inte slå in dem direkt!

Här visar vi också en översättning från kodsiffrorna till motsvarande funktioner hos processorn Z80.

```
1 REM LPRND.LPRNDLP .TAN
2 IF PEEK 16526+256*PEEK 16527
9=31744 THEN GOTO 5
3 PRINT"MINNET ÄR KLART"
4 STOP
5 FOR I=0 TO 112
6 POKE 31744+I,PEEK (16361+I)
7 NEXT I
8 POKE 31900,63
9 POKE 31887,201
10 POKE 16526,95
11 POKE 16527,179
12 POKE 16528,6
50 LET T=A*PI/180
60 FOR Y=0 TO 255
70 LET Y=I
80 GOSUB 9980
90 NEXT Y
100 FOR X=0 TO 255
110 LET Y=128+I
120 GOSUB 9980
140 NEXT X
150 NEXT I
160 FOR X=0 TO 128
170 LET Y=128/X**2/128
180 GOSUB 9980
190 NEXT X
200 FOR X=64 TO 255
210 LET Y=128+64**2/128
215 GOSUB 9980
220 NEXT X
230 FOR Y=0 TO 128+64**2/128
240 LET X=64
250 GOSUB 9980
260 NEXT Y
270 FOR Y=128 TO 0 STEP -.1
280 LET X=64+.5*(128-Y)*SIN ((Y
-128)/41)
290 GOSUB 9980
300 NEXT Y
310 FOR Y=128 TO 255 STEP .1
320 LET Y=128+(64+.5*(128-X)*SI
N ((X-128)/41))**2/128
330 GOSUB 9980
340 NEXT X
1000 GOTO 9998
9980 REM PLOT (X,Y) I A$
9981 IF X<0 OR X>255 OR Y<0 OR Y
>1855 THEN RETURN
9982 LET C=1+INT (X/8)
9983 LET R=256-INT Y
9984 POKE 16388,CODE A$(C,R)
9985 POKE 16527,2**(8+C-INT (X-1)
9986 LET A$(C,R)=CHR$ (USR 16514
)
9987 RETURN
9988 REM PRINT A$ 8 LINJER PR EN
GANG
9990 FOR I=0 TO 256 STEP 8
9991 FOR J=1 TO 32
9992 FOR K=1 TO 8
9993 POKE 32255+K+8*(J-1),CODE A
$(J,K)
9994 NEXT K
9995 NEXT J
9996 FOR H=0 TO 31
9997 POKE 16414+H,H
9998 NEXT H
9999 LET HPRINT=USR 31744
```

High resolution on the high seas — that destroyer will blast you out of the water if you surface. Even if you don't Grant Passmore's mines stand between you and safety.

CAN YOU navigate your submarine through a narrow channel without hitting the seabed or the surface? Fine. But can you also guide the craft through a lethal enemy minefield without getting blown up? Enough to give even the most experienced submariner a sinking feeling.

There are six skill levels and four game speeds. Each skill level increases the number of mines. The skill level and game speed both affect your score.

The program is in two sections, a 2K machine-code routine and a Basic listing. The game is contained in the machine code. The Basic sets the game speed, the number of mines and after each game gives a report and calculates your score. N/L is Newline.

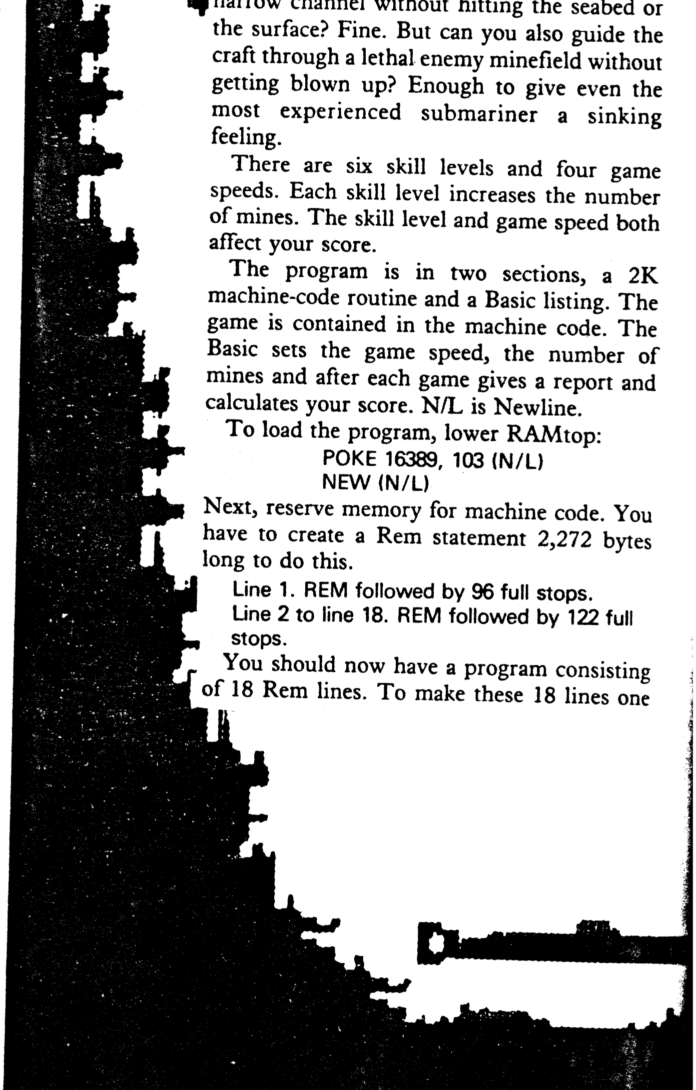
To load the program, lower RAMtop:

POKE 16389, 103 (N/L)
NEW (N/L)

Next, reserve memory for machine code. You have to create a Rem statement 2,272 bytes long to do this.

Line 1. REM followed by 96 full stops.
Line 2 to line 18. REM followed by 122 full stops.

You should now have a program consisting of 18 Rem lines. To make these 18 lines one



Program 1. The Basic listing.

```
4 SAVE "SEA MINES"
5 RAND
6 CLS
8 PRINT AT 0,0;"    MINES   SEA                       G.S.P. COMPUTER SOFTWARE"
10 LET SC=0
12 LET TSC=0
15 LET S$="000000"
18 LET R$=S$
20 LET SU=5
40 PRINT AT 8,3;"SELECT INITIAL SKILL LEVEL";C$;C$,TAB 7;"1. LIEUTENANT";TAB 7;"2. LIEUT. COMMANDER";TAB 7;"3. COMMANDER";TAB 7;"4. CAPTAIN";TAB 7;"5. COMMODORE";TAB 7;"6. ADMIRAL";C$;C$;TAB 10;"PRESS KEY 1 TO 6"
60 LET K$=INKEY$
61 PRINT AT 19,20;"■ TO ■"
62 FOR F=1 TO 3
63 NEXT F
64 PRINT AT 19,20;"1 TO 6"
65 IF K$="" THEN GOTO 60
70 IF CODE K$<29 OR CODE K$>34 THEN GOTO 60
72 GOSUB 1000
75 LET A=CODE K$-28
80 PRINT AT 8,3;"SELECT GAME SPEED.";C$;C$;TAB 7;"1. VERY FAST";TAB 7;"2. FAST";TAB 7;"3. AVERAGE";TAB 7;"4. SLOW";C$;C$;C$;C$;TAB 10;"PRESS KEY 1 TO 4"
81 IF INKEY$<>"" THEN GOTO 81
82 LET K$=INKEY$
83 PRINT AT 19,20;"■ TO ■"
84 FOR F=1 TO 3
85 NEXT F
87 PRINT AT 19,20;"1 TO 4"
89 IF K$="" THEN GOTO 82
90 IF CODE K$<29 OR CODE K$>32 THEN GOTO 82
95 LET B=CODE K$-28
100 GOSUB 1000
105 GOSUB 1500
109 IF TSC<>0 THEN GOTO 200
110 IF TSC=0 THEN PRINT AT 10,15-(LEN D$/2),D$;C$;C$;C$;TAB 5;"YOU HAVE 5 SUBMARINES";C$;C$;TAB 11;"[illegible]"
135 GOSUB 2000
140 GOSUB 1000
210 POKE 17624,A+2
215 POKE 16434,(INT (RND*254))+2
220 POKE 16910,(B*2)
230 LET X=USR 18400
240 IF A*B*RND<3 THEN GOTO 270
245 POKE 26929,156
250 POKE 26962,156
260 POKE 26995,156
270 LET X=USR 18433
275 LET ST=PEEK 16417
280 IF ST=3 THEN GOTO 500
281 LET A=A+1
282 IF A=7 THEN GOTO 330
284 GOSUB 1500
285 PRINT AT 8,5;"YOU HAVE ESCAPED FROM";TAB 5;"THE MINEFIELD";C$;TAB 5;"GAINING PROMOTION TO";TAB 5;D$;C$;C$;C$
300 GOTO 340
330 PRINT AT 8,5;"YOU HAVE ESCAPED FROM ";TAB 5;"THE MINEFIELD ADMIRAL";C$;C$;C$;C$;C$;C$;
337 LET A=A-1
340 LET FAC=4
345 GOSUB 4010
350 PRINT TAB 5;"YOU SCORED : ";S$;TAB 5;"TOTAL SCORE: ";R$
351 GOSUB 2120
353 GOSUB 1000
355 IF SU=0 THEN GOTO 5000
360 GOTO 900
500 LET PO=PEEK 16592+(256*PEEK 16593)
505 LET SU=SU-1
507 IF SU=0 THEN GOTO 5000
510 LET FA=INT ((PO-26731)/33)
520 LET FA=FA+2
530 IF PO<26761 THEN PRINT AT 8,15-(LEN D$/2),D$;C$;C$;TAB 4;"YOU BROKE SURFACE.";C$;TAB 4;"THE FRIGATE DESTROYED          YOUR SUBMARINE";C$;C$;
532 IF PO<26761 THEN LET FAC=3
535 IF PO<26761 THEN GOTO 650
539 LET F=158
540 IF PEEK (PO+31)<>F OR PEEK (PO+196)<>F OR PEEK (PO+200)<>F OR PEEK (PO+165)<>F OR PEEK (PO+68)<>F OR PEEK (PO+166)<>F THEN GOTO 600
550 LET FAC=2
570 PRINT AT 8,15-(LEN D$/2);D$,C$;C$;TAB 5;"YOU HIT A MINE AT A";C$;TAB 5;"DEPTH OF ";FA;" FATHOMS";C$;C$;
580 GOSUB 650
600 PRINT AT 6,15-(LEN D$/2);D$;C$;C$;TAB 6;"YOU HIT THE SEABED AT";C$;TAB 6;"A DEPTH OF ";FA;" FATHOMS";C$;C$;
605 LET FAC=1
650 GOSUB 4000
660 PRINT TAB 5,C$,"YOU SCORED : ";S$;;TAB 5;"TOTAL SCORE: ";R$;C$;TAB 5;"REMAINING VESSELS : ";SU
665 GOSUB 2000
670 GOSUB 1000
675 IF SU=0 THEN GOTO 5000
677 IF B<3 THEN GOTO 900
680 GOTO 200
900 PRINT AT 8,5;"PRESENT GAME SPEED : ",B;C$;C$;TAB 5;"DO YOU WANT TO CHANGE ",TAB 5;"THE GAME SPEED ?";C$;C$;C$;C$;C$;C$;TAB 10,"PRESS KEY YES OR NO"
910 LET K$=INKEY$
915 PRINT AT 19,20,"■ES OR ■O"
916 FOR F=1 TO 2
917 NEXT F
919 PRINT AT 19,20;"YES OR NO"
920 IF K$="" THEN GOTO 910
925 IF K$<>"N" AND K$<>"Y" THEN GOTO 910
930 IF K$="N" THEN GOTO 970
940 IF INKEY$<>"" THEN GOTO 940
950 GOSUB 1000
960 GOTO 80
970 GOSUB 1000
980 GOTO 200
999 STOP
1000 PRINT AT 5,0;C$,C$;C$;C$;C$;C$;C$;C$;C$;C$;C$;C$;C$;C$;C$;C$;C$
1010 RETURN
1500 IF A=1 THEN LET D$="LIEUTENANT"
1510 IF A=2 THEN LET D$="LIEUTENANT COMMANDER"
1520 IF A=3 THEN LET D$="COMMANDER"
1530 IF A=4 THEN LET D$="CAPTAIN"
1540 IF A=5 THEN LET D$="COMMODORE"
1550 IF A=6 THEN LET D$="ADMIRAL"
1560 RETURN
2000 GOSUB 2270
2010 IF RND<.5 THEN GOTO 2050
2020 LET K$=T$(1+((RN-1)*24) TO (RN*24))
2030 LET X=16987
2040 GOTO 2070
2050 LET K$=T$(73+((RN-1)*24) TO 72+(RN*24))
2060 LET X=17011
2070 GOSUB 2240
2080 LET K$=T$(289+((RN-1)*9) TO 288+(RN*9))
2090 LET X=17083
2100 GOTO 2290
2120 GOSUB 2270
2130 IF RND<.5 THEN GOTO 2170
2140 LET K$=T$(145+((RN-1)*24) TO 144+(RN*24))
2150 LET X=17035
2160 GOTO 2190
2170 LET K$=T$(217+((RN-1)*24) TO 216+(RN*24))
2180 LET X=17059
2190 GOSUB 2240
2200 LET K$=T$(316+((RN-1)*9) TO 315+(RN*9))
2210 LET X=17092
2220 GOTO 2290
2240 FOR F=1 TO 24
2250 POKE (X+F),CODE K$(F)
2260 NEXT F
2270 LET RN=(INT (RND*3))+1
2280 RETURN
2290 FOR F=1 TO 9
2300 POKE (X+F),CODE K$(F)
2310 NEXT F
2320 RETURN
4000 LET SC=INT (A*(5-B)*(100/FA)+FAC)*3
4005 IF FAC<>4 THEN GOTO 4012
4010 LET SC=INT (A*(5-B)*FAC)*24
4012 LET TSC=TSC+SC
4015 LET S$=O$(3 TO 6-(LEN (STR$ SC)))+STR$ SC
4020 LET R$=O$(1 TO 6-(LEN (STR$ TSC)))+STR$ TSC
4030 RETURN
5000 PRINT AT 7,10;C$;C$;TAB 5;"ALL SUBMARINES LOST";C$;C$;TAB 5;"YOU SCORED : ";R$
5001 LET H$=O$(1 TO (6-(LEN STR$ HSC)))+STR$ HSC
5003 PRINT C$;TAB 5;"HI - SCORE : ";H$
5004 IF HSC<TSC THEN PRINT C$;C$;TAB 6;"A NEW HI-SCORE.";TAB 11;"[illegible]"
5005 IF HSC<TSC THEN LET HSC=TSC
5006 FOR F=1 TO 15
5007 PRINT AT 7,10;"GAME OVER"
5009 FOR A=1 TO 2
5011 NEXT A
5013 PRINT AT 7,10;"[illegible]"
5015 NEXT F
5025 PRINT AT 21,4;"PRESS [illegible] TO REPLAY"
5026 FOR F=1 TO 2
5027 NEXT F
5029 PRINT AT 21,4;"PRESS NEWLINE TO REPLAY"
5030 IF INKEY$<>CHR$ 118 THEN GOTO 5025
5035 GOSUB 1000
5040 GOTO 8
```

The hexadecimal loader.

```
9000 PRINT "START ADDRESS?"
9010 INPUT X
9015 CLS
9020 LET A$=""
9025 PRINT X;":";
9030 FOR A=1 TO 8
9035 IF A$="" THEN INPUT A$
9040 IF A$="S" THEN STOP
9045 PRINT A$( TO 2);" ";
9050 POKE X,16*CODE A$+CODE A$(2)-476
9060 LET A$=A$(3 TO )
9070 LET X=X+1
9080 NEXT A
9090 PRINT
9095 GOTO 9025
9200 PRINT "START ADDRESS?"
9210 INPUT X
9215 CLS
9220 PRINT X;":";
9225 FOR A=1 TO 8
9230 LET A$="  "
9240 LET B=PEEK X
9345 LET A$(1)=CHR$ (INT (B/16)+28)
9250 LET A$(2)=CHR$ (B-(INT (B/16)*16)+28)
9260 PRINT A$;" ";
9270 LET X=X+1
9280 NEXT A
9285 PRINT
9290 GOTO 9220
```

Sea Mines hexadecimal dump.

```
16514:3E 1E ED 47 DD 21 81 02
16522:C9 3E 08 ED 47 DD 21 A7
16530:40 C9 21 00 67 0E C0 06
16538:20 36 9E 23 10 FB 36 C9
16546:23 0D 20 F3 C9 21 DF E6
16554:11 21 00 F3 0E FE 06 16
16562:10 FE 06 C0 ED 78 D3 FF
16570:19 CD CF 40 05 C2 B6 40
16578:CD 92 02 CD 20 02 DD 21
16586:A7 40 C3 A4 02 E9 33 73
16594:2A D0 40 11 05 41 1A 77
16602:01 1F 00 09 13 1A 77 23
16610:23 13 1A 77 09 13 0E 04
16618:06 05 1A 77 23 13 10 FA
16626:C5 01 1C 00 09 C1 0D 20
16634:EF 1A 77 23 23 23 23 13
16642:1A 77 C9 20 97 24 A2 1E
16650:1E 1E A2 9B 1E 1E 1E 32
16658:9B 1E 1E 1E 8F 1E 1E 1E
16666:1E 1E 97 20 2A D0 40 11
16674:21 00 ED 52 3E 9E 77 01
16682:1F 00 09 77 03 03 03 09
16690:77 23 23 77 23 77 CD D2
16698:40 C9 2A D0 40 11 C5 00
16706:19 3E 9E 77 23 77 23 77
16714:11 1E 00 19 77 23 23 23
16722:23 77 CD D2 40 C9 2A D0
16730:40 2B 3E 9E 77 11 1F 00
16738:19 77 23 23 77 19 77 13
16746:13 06 04 19 77 10 FC 23
16754:23 23 23 77 CD D2 40 C9
16762:2A D0 40 23 3E 9E 77 11
16770:1F 00 19 77 23 23 77 11
16778:23 00 19 77 1E 1B 06 04
16786:19 77 10 FC 2B 2B 2B 2B
16794:77 CD D2 40 C9 2A D0 40
16802:11 C1 41 CD D6 40 CD 0C
16810:42 2A D0 40 11 DA 41 CD
16818:D6 40 CD 0C 42 2A D0 40
16826:11 F3 41 CD D6 40 C9 00
16834:08 98 07 9A 3C 0D 07 94
16842:0D 98 01 1C 01 01 95 95
16850:83 93 38 93 38 01 0B 00
16858:9E 9E 2A 08 2A 08 0B 00
16866:2A A5 00 0E 0B 0E B8 0B
16874:2A 09 B5 08 BE 08 B8 0B
16882:2A 9E 9E 9E 9E 9E 9E 9E
16890:9E 9E 9E 9E 9E 9E 9E 9E
16898:9E 9E 9E 9E 9E 9E 9E 9E
16906:9E 9E 11 00 1E 18 7A 83
16914:20 FB C9 3A 7B 40 FE 01
16922:20 0A 2A D0 40 2B 22 D0
16930:40 CD 7A 41 FE 03 20 0A
16938:2A D0 40 23 22 D0 40 CD
16946:58 41 3A 7C 40 FE 01 20
16954:0D 2A D0 40 11 21 00 19
16962:22 D0 40 CD 1E 41 FE 02
16970:C0 2A D0 40 11 21 00 C6
16978:00 ED 52 22 D0 40 CD 3C
16986:41 C9 99 36 22 22 2E 2E
16994:0E 22 22 17 17 6B 1E 17
17002:22 22 22 6C 17 17 22 01
17010:1E 17 9E 08 08 96 96 96
17018:97 8A 8A 32 32 32 A2 A2
17026:A2 A2 32 A2 A2 A2 8A AA
17034:1E 1E 9E 9E 1D 96 97 97
17042:97 97 0B 9E 1D 96 97 A2
17050:97 97 0B 9E 9E 0B 97 A2
17058:A2 97 9E 9E 9E 99 03 03
17066:22 22 17 22 99 9E 9E 2E
17074:9E 9E 2E 9E 2E 22 0E 2E
17082:22 9E 0B 0B 97 19 32 97
17090:A2 32 32 9E 9E 2E 22 39
17098:8C 17 17 30 0E 22 A2 8B
17106:32 11 0B 00 3E 1E 21 41
17114:7F 0E 04 06 16 77 23 10
17122:FC 19 0C 20 F6 3E 1E 21
17130:6D 68 06 1C 77 23 10 FC
17138:06 90 21 30 6D 11 1F 00
17146:77 19 77 23 23 10 F9 21
17154:61 73 06 60 11 1D 00 77
17162:19 77 23 23 23 23 10 F7
17170:21 92 79 06 30 11 1B 00
17178:77 19 77 11 06 00 19 10
17186:F4 21 AB 7C 0E 18 11 19
17194:00 77 19 77 11 08 00 19
17202:10 F4 21 38 7E 06 0C 11
17210:17 00 77 19 77 11 0A 00
17218:19 10 F4 21 E6 7A 11 21
17226:00 06 22 77 19 10 FC 21
17234:7E 7D 11 1F 00 06 0E 77
17242:23 23 77 19 10 F9 21 41
17250:7E 06 08 11 19 00 77 23
17258:77 23 77 23 23 23 23 77
17266:23 77 23 77 19 10 EF 18
17274:0E 06 09 1A 77 D5 11 21
17282:00 19 D1 13 10 F5 C9 21
17290:7C 43 3E 18 77 21 18 6A
17298:11 74 42 CD 7B 43 23 11
17306:8C 42 CD 7B 43 11 74 42
17314:CD 7B 43 23 11 8C 42 CD
17322:7B 43 11 74 42 CD 7B 43
17330:23 11 74 42 CD 7B 43 21
17338:37 6A 11 5C 42 CD 7B 43
17346:2B 11 A4 42 CD 7B 43 11
17354:5C 42 CD 7B 43 2B 11 A4
17362:42 CD 7B 43 11 5C 42 CD
17370:7B 43 2B 11 5C 42 CD 7B
17378:43 21 66 7A 11 5C 42 CD
17386:7B 43 21 68 7A 11 74 42
17394:CD 7B 43 21 7C 43 3E 09
17402:77 21 0F 7D 11 BC 42 CD
17410:7B 43 21 18 7E 11 BC 42
17418:CD 7B 43 21 1E 7D 11 BC
17426:42 CD 7B 43 21 29 7E 11
17434:BC 42 CD 7B 43 21 1F 7E
17442:11 C5 42 CD 7B 43 21 1A
17450:7D 11 C5 42 CD 7B 43 21
17458:2D 7E 11 C5 42 CD 7B 43
17466:21 25 7D 11 C5 42 CD 7B
17474:43 C3 61 44 1A 77 13 D5
17482:11 21 00 19 D1 06 02 1A
17490:77 23 13 1A 77 13 D5 11
17498:20 00 19 D1 10 F1 C9 21
17506:DF 7E 11 CE 42 CD 45 44
17514:21 E2 7E 11 CE 42 CD 45
17522:44 21 F0 7E 11 CE 42 CD
17530:45 44 21 DE 7D 11 CE 42
17538:CD 45 44 21 E5 7D 11 CE
17546:42 CD 45 44 C9 CD BB 02
17554:7C FE FF C8 7D FE F7 20
17562:05 FE EF C0 18 0D 7C FE
17570:DF C0 3A 7B 40 3D C8 32
17578:7B 40 C9 7C FE DF 20 09
17586:3A 7C 40 3D C8 32 7C 40
17594:C9 FE EF 20 0E 3A 7C 40
17602:3C FE 03 C8 32 7C 40 C9
17610:FE F7 C0 3A 7B 40 3C FE
17618:04 C8 32 7B 40 C9 01 0C
17626:31 69 21 31 69 22 DA 44
17634:3E 0C 32 D9 44 3A D8 44
17642:3D C8 32 D8 44 3A D9 44
17650:3D 28 E7 32 D9 44 2A DA
17658:44 11 21 00 0E 03 7E FE
17666:9E 20 11 19 10 F6 2A DA
17674:44 3E 99 77 19 3E 03 77
17682:19 3E 99 77 06 03 CD [illegible]
17690:45 11 32 00 C6 00 ED [illegible]
17698:16 00 5D 2A DA 44 19 [illegible]
17706:DA 44 10 EA 18 EF 9E [illegible]
17714:E8 45 19 10 F7 23 23 7E
17722:FE 2A 32 40 54 5D 19 [illegible]
17730:29 19 19 16 29 19 22 [illegible]
17738:40 C9 29 29 16 22 32 40
17746:C9 40 11 20 03 19 7E FE
17754:9E C2 E6 45 19 2B 13 [illegible]
17762:06 7E FE 9E C2 E8 45 19
17770:10 F7 C6 00 ED 52 C6 [illegible]
17778:ED 52 23 23 06 03 7E FE
17786:9E C2 E8 45 23 10 F7 [illegible]
17794:19 7E FE 9E C2 E8 45 [illegible]
17802:3A 7C 40 FE 01 CA C1 [illegible]
17810:2A D0 40 11 21 00 C6 [illegible]
17818:ED 52 7E FE 9E C2 E8 [illegible]
17826:19 2B 2B 7E FE 9E C2 [illegible]
17834:45 19 23 7E FE 9E C2 [illegible]
17842:45 23 23 06 02 7E FE [illegible]
17850:C2 E6 45 23 10 F7 C9 [illegible]
17858:D0 40 11 C5 00 19 06 [illegible]
17866:7E FE 9E C2 E6 45 23 [illegible]
```

SEA MINES

ZX-81

line enter the following direct commands:

POKE 16514 ,118 (N/L)
POKE 16515 ,118 (N/L)
POKE 16511 ,226 (N/L)
POKE 16512 ,8 (N/L)

Input line 4. Save "Sea Mine (Inv S)".

LIST 4 (N/L)
POKE 16419 ,4 (N/L)

Then enter the hexadecimal loader routine.

GOTO 9000 (N/L)

On the prompt enter 16514 then key in the machine-code listing. When you have finished, Save the machine code on tape a couple of times. To test the machine code, add the following lines:

10 RAND USR 18400
20 RAND USR 18433
30 STOP
GOTO 10 (N/L)

You should now have a picture of a submarine in a deep trench with a frigate waiting on the surface and the seabed climbing steeply up either side of the screen. There should also be a scattering of mines above the submarine.

Press key 6 — the down cursor — and the submarine should slowly move down to the sea bed and be destroyed. The program will then return to Basic with report code 9/30.

If it does not run correctly then lower RAMtop and reload the game from tape. Use the program in the hexadecimal loader routine starting at 9200 to check your machine-code listing against the published listing. Correct any mistakes you find and then rerun the test program. When the machine code has been debugged Save it. Delete all the lines from line 10 onwards.

Variables:

CLEAR (N/L)

Enter program 2 to assign the characters to T$. Then Run the program and enter the numbers in listing 1.

GOTO 10 (N/L)

Delete all the lines from line 10 onwards.

Never Run or Clear the computer, use Goto instead of Run.

Finally, enter the Basic listing. To Save the game and make it autostart when loaded do the following:

1. Enter GOTO 1 — do not press N/L yet
2. Start the tape recorder
3. Press N/L

Before you load the game from tape again you will have to lower RAMtop.

POKE 16389, 103 (N/L)
NEW (N/L)
LOAD ""
start tape
(N/L)

Thanks to Richard Taylor for his high-resolution routine. ■

```
17874:F7 11 1D 00 19 7E FE 9E
17882:C9 E8 45 23 23 23 7E
17890:FE 9E C2 E8 45 C9 FE C9
17898:20 05 3E 02 32 8E 44 C9
17906:3E 03 32 8E 44 C9 CD BB
17914:02 7C FE FF C8 7D FE F7
17922:20 05 FF 5F C9 18 0D
17930:FE DF C0 3A 7B 40 3D C9
17938:3E 7B 40 C9 7C FE DF 20
17946:06 3A 7C 40 3D C8 32 7C
17954:40 C9 FE EF 20 05 3A 7C
17962:40 3C FE 03 C8 32 7C 40
17970:C9 FE F7 C0 3A 7B 40 3C
17978:FE 04 C8 32 7B 40 C9 21
17986:55 7D 22 D8 40 21 02 02
17994:02 7B 40 CD 94 49 03 3A
18002:7B 40 FE 01 CA DA 46 FE
18010:02 CA 6A 47 3A 7C 40 FE
18018:01 CA A2 46 2A D8 40 11
18026:20 00 C5 00 ED 52 7C 40
18034:9E C2 C8 47 1B 06 02 19
18042:7E FE 9E C2 C8 47 23 23
18050:7E FE 9E C2 C8 47 10 EF
18058:23 7E FE 9E C2 C8 47 23
18066:13 7E FE 9E C2 C8 47 23
18074:C8 47 19 10 F7 C9 00 00
18082:2A D8 40 11 22 00 19 7E
18090:FE 9E C2 C6 47 19 19 10
18098:06 05 7E FE 9E C2 C6 47
18106:19 10 F7 11 46 00 06 00
18114:ED 52 06 03 7E FE 9E C2
18122:C8 47 23 10 F7 11 1E 00
18130:19 7E FE 9E C2 C8 47 C9
18138:3A 7C 40 FE 01 CA 29 47
18146:2A D8 40 11 22 00 C6 00
18154:ED 52 7E FE 9E C2 C8 47
18162:11 1F 00 06 02 19 7E FE
18170:9E C2 C8 47 23 23 7E FE
18178:9E C2 C8 47 10 EF 23 23
18186:7E FE 9E C2 C8 47 2B 2B
18194:19 13 13 06 04 7E FE 9E
18202:C2 C8 47 19 10 F7 C9 00
18210:00 C2 C6 47 19 10 F7 2A
18218:D8 40 11 20 00 19 7E FE
18226:9E C2 C6 47 19 2B 13 06
18234:04 7E FE 9E C2 C8 47 19
18242:10 F7 7E FE 9E C2 C8 47
18250:23 23 06 03 7E FE 9E C2
18258:C8 47 23 10 F7 19 2B 7E
18266:FE 9E C2 C8 47 2B 2B 2B
18274:2B 7E FE 9E C2 C8 47 C9
18282:3A 7C 40 FE 01 CA A1 47

18290:2A D8 40 11 21 00 C6 00
18298:ED 52 7E FE 9E C2 C8 47
18306:19 2B 2B 7E FE 9E C2 C8
18314:47 19 23 7E FE 9E C2 C8
18322:47 23 23 06 02 7E FE 9E
18330:C2 C8 47 23 10 F7 C9 2A
18338:D8 40 11 C5 00 19 0D 03
18346:7E FE 9E C2 C8 47 23 10
18354:F7 11 1D 00 19 7E FE 9E
18362:C2 C8 47 23 23 23 23 7E
18370:FE 9E C2 C8 47 C9 FE C9
18378:9E C8 FE C9 20 06 3E 02
18386:32 50 46 C9 3E 03 32 50
18394:46 C9 02 02 01 00 21 55
18402:7D 22 D8 40 3E 01 32 60
18410:46 21 02 02 22 7B 40 CD
18418:94 49 CD D5 42 CD D2 49
18426:CD DC 44 CD 48 49 C9 CD
18434:BB 49 CD BB 02 7C FE FF
18442:28 F8 3E 02 32 DF 47 2A
18450:7B 40 22 DC 47 3E 01 32
18458:DE 47 3A DE 47 FE 02 28
18466:03 CD 6F 44 CD 0C 42 ED
18474:5B DC 47 3A 7B 40 BB 20
18482:9E 3A 7C 40 3D C8 32 7C
18490:02 32 DE 47 3A DF 47 3D
18498:32 DF 47 20 D5 CD B5 02
18506:7C FE FD 20 05 3E 02 32
18514:7B 40 CD 51 46 3A 50 46
18522:FE 02 28 04 CD 15 46 C9
18530:0C 42 CD 0C 42 3A 50 46
18538:FE 01 28 9E 06 04 3A 0E
18546:42 4F 61 10 FD 4F 32 0E
18554:CD 0C 42 CD 9F 41 CD 0C
18562:42 CD 52 49 CD 94 49 CD
18570:D3 42 3E 03 32 21 40 C9
18578:06 04 3A 0E 42 4F 61 10
18586:FD 32 0E 42 CD 0C 42 CD
18594:52 49 CD 94 49 CD 9F 41
18602:3E 04 32 21 40 C9 7B 40
18610:3E 05 32 7C 40 3A 7B 40
18618:FE 01 28 27 2A D0 40 23
18626:3E 07 06 04 7E 23 77 2B
18634:23 23 23 3E 9E 77 11
18642:24 00 19 0D 20 EC CD 0C
18650:42 3A 7C 40 3D C8 32 7C
18658:40 18 D9 2A D0 40 2B 0E
18666:07 06 04 7E 2B 77 23 23
18674:10 F9 2B 3E 9E 77 11 1E
18682:00 19 0D 20 EC CD 0C 42
18690:3A 7C 40 3D C8 32 7C 40
18698:18 D9 9E 9E 99 9E 9E 9E

18706:2A 99 9E 9E 9E 2A 99 9E
18714:9E 9E 2A 9E 9E 9E 2A 9E
18722:22 9E 9E 9E 2A 64 95 9E
18730:9E 0A 64 A2 9E 9E 17 1E
18738:97 9E 9E 17 1E 63 1E 22
18746:1E 1E 1E 32 1E 1E 1E 1E
18754:A2 1E 1E 1E 1E 1E 1E 0C
18762:49 21 0F 67 9E 0C 06 05
18770:1A 77 13 23 10 FA D5 11
18778:1C 00 19 D1 0D 20 EF C9
18786:18 00 04 0D 00 F8 39
18794:2A 25 00 32 2E 33 2A 00
18802:0B 76
```

Program 2.

```
10 DIM T$(342)
20 FOR N=1 TO 342
30 INPUT B
40 LET T$(N)=CHR$ B
50 PRINT CHR$ B;
60 NEXT N
```

Listing 1.

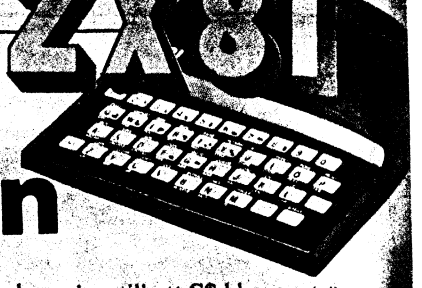
```
153 54 34 34 46 46
14 34 34 23 23 139
30 23 34 34 34 140
23 23 34 1 30 23
158 153 153 14 34 34
14 46 34 34 34 140
23 140 34 23 23 23
34 23 139 30 17 23
158 158 158 153 54 3
34 34 23 140 34 34
23 23 23 139 30 23
22 139 30 30 139 23
29 29 29 151 162 151
151 162 151 151 162 50
30 30 151 143 50 162
151 151 50 30 50 50
158 11 11 150 150 150

151 138 138 50 50 50
162 162 162 162 50 162
162 162 138 170 30 30
158 158 158 11 150 151
151 151 151 162 162 162
162 143 143 30 30 30
30 50 50 50 162 50
158 158 158 11 151 162
162 151 151 138 50 162
11 158 158 29 151 151
162 151 11 158 158 158
158 158 158 158 11 150
151 151 151 11 158 151
158 151 151 151 11 158
150 151 151 150 158 158
158 158 29 150 151 151
151 151 11 158 29 150
151 162 151 151 11 158
158 11 151 162 162 151
158 46 14 34 54 15
158 158 46 158 158 153
3 23 34 23 23 34
46 34 54 153 158 158
158 158 158 153 3 3
34 34 23 34 153 158
158 46 158 158 46 158
46 34 14 46 34 158
158 158 158 158 158 46
46 46 34 34 14 46
46 34 23 23 34 46
158 158 158 46 46 158
11 11 151 25 50 151
162 50 50 44 19 55
138 50 50 30 50 30
29 29 29 5 62 143
143 30 30 158 158 46
34 57 140 23 23 46
153 14 34 34 140 23
23 17 155 158 15 34
46 29 20 30 23 30
```

The article accompanying the 2X81 Sea-mines program in Sept. neglected to mention three variables which need to be assigned a value directly from the keyboard. When you have finished keying in the program enter the following statements:

```
LET C$="32 spaces"
LET O$="0000000"
LET HSC=7000
```

last column:

43,7B,D5,1A,11,21,44,45,CD,42,CE,02,28,FE
32,09,40,40,C9,FE,0C,44,44,44,DA,FE,0A,77,3B
5422,C2,7E,29,32,40,FE,06,19,00,FE,2B,C9,45
00,45,E8,E8,9E,2A,03,10



"Sifferjakt" i ny version

★ I ett tidigare nummer presenterade Gunnar Farm en "sifferjakt" för ZX81 och ansluten a/d-omvandlare. Här kommer en förbättrad version av programmet som också går att köra på ZX80.

★ Författaren visar också sin lösning på ett störningsproblem som kan uppstå med extra minne anslutet.

Av BENGT ANDERSSON

▶ Som programmet *Sifferjakt* i RT *1982 nr 11* är skrivet går det inte att köra på **ZX80.** Det blir dock möjligt om man själv styr bildastringen. Det visar sig att programmet på så sätt blir enklare samt även ger andra fördelar och alltså kan vara intressant också för ägare av ZX81. Den enda svårigheten när man skriver denna typ av program är att få tiderna att stämma.

Apropå tider: Klockoscillatorn i det visade kopplingsschemat för a/d-omvandlaren ger alldeles för låg frekvens med angivet kondensatorvärde (C1, 1 nF). Frekvensen blir i praktiken ca 340 kHz. Värdet är visserligen okritiskt men för lågt för att man skall kunna konstruera om programmet på det sätt jag gjort. Prov har visat att kapacitansen bör vara 500–600 pF för att frekvensen skall bli den rätta.

Bättre tidtagning utan felläsning

Så till programmet: Tidtagningen enligt RT 11 kan ge fel resultat. Värdet **FRAMES** kan nämligen bli noll. Sannolikheten är visserligen liten, men felet uppträder slumpvis och kan verka förbryllande. Ett säkrare sätt än att läsa av **FRAMES** två gånger är att först nollställa (med **POKE** eller enklast i maskinspråkrutinen). Men det går att göra det ännu enklare:

När man själv sköter räkningen av antalet delbilder är man inte låst till **Sinclairs** räknesekvens (ner från 0 till 32 769) utan kan i stället räkna upp från 0 till 65 535. En extra fördel är att man kan mäta tider upp till ca 22 minuter, om nu någon skulle behöva det.

Vi kan lägga in antalet delbilder i BC-registret vid returen. Vi måste då visserligen lagra resultatet av spelet i en särskild minneadress, som sedan får läsas av med **PEEK**, men det hela blir enklare så.

Det är också enklast att lägga ut delresultaten av uppräkningen på stacken, så slipper vi att återställa **FRAMES** till ett tillåtet värde vid returen. Vidare kan det vara bra att kunna stoppa programmet med **BREAK**, ifall man av misstag skulle köra i gång utan att ha a/d-omvandlaren ansluten. (Inte så trevligt att behöva dra ur kontakten så att programmet faller.)

Programmet kan köras med 3 K minne. Här följer nu en kort beskrivning av maskinkoddelen:

Maskinkod styr bild

Koden för "1" läggs in i A'. Omvandlaren startas. (Min omvandlare tar upp ett helt K från 20 480; adresserna är olika i maskinkoddelen och basicdelen. Detta är lätt att ändra, men observera att avläsningsordningen är omkastad i förhållande till Gunnar Farms program.) Tiden läggs ut på stacken (första gången Ø). **DFILE** hämtas in och läggs ut på stacken. I adress 16 533 anropas subrutin 3 910 i basictolken, som kontrollerar om **BREAK** har utförts.

Samtidigt startar bildsynkpulsen. Därefter följer avläsning och beräkning av markörens position. Observera, hur avläsningen i vertikalled görs med **RLD!** Samtidigt sker division med 16 (bit 4–7 i ackumulatorn är redan nollställda). **RLD** startar dessutom omvandlaren för ny avläsning på samma adress, men det saknar betydelse.

I adress 16 567 startas kontrollen av beräknad position. Om spelet skall avbrytas läggs vid misslyckande Ø och vid lyckad jakt innehållet i A' (det vill säga koden för "9") in i adress 16 514, och antalet delbilder hämtas in från stacken. Skall spelet fortsätta läggs prickens position ut på stacken, samtidigt som **DFILE** hämtas in, och bildgenereringen börjar. När bilden är färdigritad hämtas prickens position in och den raderas. Därefter hämtas tiden in, ökas med ett, och allt upprepas.

Beträffande tidschemat har jag följt ZX81:s tider med två undantag: Avståndet från bildsynk till första linjesynkpuls har gjorts en klockcykel längre, som i ZX80:s heltalsbasic. Den första linjen under texten är i ZX81 fyra klockcykler för kort, vilket gör den tydligt mörkare. En **NOP** har lagts in (16 594).

Bildsynkens längd och placering verkar däremot inte vara särskilt kritisk. Om bilden antas bestå av 312 linjer, var och en på 207 klockcykler, så borde bildsynken komma 25 klockcykler tidigare och vara 27 klockcykler längre (förutsatt att jag räknat rätt), om den antas vara 6 linjer plus en linjesynkpuls.

Basicdelen med långa rader

Basicdelen, rad 10–40. Sinclair tillåter obegränsat långa rader, och man skall ju någon gång använda det! Man sparar minne på det viset, men det medför vissa nackdelar. Den allvarligaste är att redigeringen försvåras. Förutsatt att man inte skriver fel anser jag ändå att det här är den bästa kompromissen. Dessutom kan överskådligheten ibland försämras, samtidigt som skrivarbetet ökar om man vill skriva in samma text två gånger. Det är orsaken till att raderna 10–30 fått vara som de är.

I rad 60 lurar vi datorn att tro att ett stort minne är anslutet. Detta har ingen annan effekt än att minnesutrymme avdelas för bildskärmen. Tvärtom kan det vara förståndigt att skriva **POKE 16389,76** innan programmet lagras på band. Vi slipper då spela in en tom bildskärm. Spar tid!

Raderna 80 och 160–230. Orsaken till att **S$** blev en strängvariabel är att flerdimensionella sådana är enklare att hantera än numeriska dito. Emellertid visade det sig att en endimensionell numrering är enklare (från 0 till 255 från övre högra hörnet till det nedre vänstra).

Det otillåtna värdet (255) kan då direkt uteslutas. Man sparar dessutom minne på att använda strängvariabler som ett slags heltalsvariabler. Talet måste dock ligga i intervallet 0–255.

Rad 320: Kravet att värdet måste vara noll visade sig orsaka problem. Detta är ett mycket strängt villkor. Spänningen måste vara mindre än 10 mV! Av out-grundlig anledning fungerade alltid programmet när man startade med **RUN** men hakade upp sig när man försökte spela igen. Att tillåta värdena ett och två ger inga problem.

Det går faktiskt att utvärdera resultatet av jakten, beräkna och skriva ut tiden i en enda programrad, rad 350.

Avkopplad trafo säkrar minnet

Så till ett annat problem: Sinclair 16 K RAM. Bortsett från det ständiga bekymret med kontakten, som kräver ständig vaksamhet, har jag drabbats av andra störningar. Efter en stunds drift ändrades innehållet slumpvis i en del minnesceller. Först misstänkte jag ett termiskt fel, men eftersom inget kunde upptäckas riktade mina blickar mot nätaggregatet.

Jag provade att driva datorn från ett kraftigt, stabiliserat nätaggregat, och felet var borta! En närmare kontroll gav till resultat att nätaggregatet ger ca 11 V. I minnet finns en diod mellan +9 V och +12 V. Varje transient från nätaggregatet som överstiger ca 13 V (dämpad enbart av förmodat dåliga elektrolytkondensatorer) leds direkt till minneskretsarna och ger störningar.

Varje gång tv:n slås till och från genereras en transient, vilket förklarar att felet oftast uppkom när jag lämnat apparaten. En kondensator på 100 nF över transformatorns sekundärlindning var tillräcklig för att avhjälpa felet.

KORTA RAPPORTER

MASKINKOD TILL SIFFERJAKT

```
14  0                       ---          8  211 255          OUT (255),A
15  62 29                   LD A,29      80 203 252          SET 7,H
17  8                       EX AF,AF'     2 253 78 40        LD C,(IY+40)
18  33 0 0                  LD HL,0       5  12              INC C
21  1 0 80            L1    LD DE,20480   6  6 25           LD B,25
24  115                     LD (DE),A     8  62 233          LD A,233
25  229                     PUSH HL      90  205 181 2       CALL 693
26  42 12 64                LD HL,(16396) 3  43              DEC HL
29  229                     PUSH HL       4  0               NOP
30  35                      INC HL        5  205 146 2       CALL 658
31  6 32                    LD B,32       8  225             POP HL
33  205 70 15               CALL 3910     9  113             LD (HL),C
36  48 72                   JR NC,L7     600 225             POP HL
38  16 254            L2    DJNZ L2       1  35              INC HL
40                          LD A,(DE)     2  24 173          JR L1
41  19                      INC DE    16604  8          L6  EX AF,AF'
42  18                      LD (DE),A     5  190             CP (HL)
43  47                      CPL           6  40 5            JR Z,L8
44  230 240                 AND 240       8  54 6            LD (HL),6
46  79                      LD C,A       10  175         L7  XOR A
47  6                       ADD HL,BC     1  24 6           JR L9
48  9                       ADD HL,BC     3  254 37     L8    CP 37
49  15                      RRCA          5  32 8           JR NZ,L10
50  15                      RRCA          7  54 151         LD (HL),151
51  15                      RRCA          9  50 130 64  L9    LD (16514),A
52  15                      RRCA         22  225             POP HL
53  79                      LD C,A        3  193             POP BC
54  9                       ADD HL,BC     4  201             RET
56  6 38                    LD B,38       5  60        L10   INC A
57  16 254            L3    DJNZ L3       6  8               EX AF,AF'
59  235                     EX DE,HL      7  24 202          JR L5
60  237 111                 RLD
62  235                     EX DE,HL
63  198 107                 ADD A,107
65  79                      LD C,A
66  9                       ADD HL,BC
67  175                     XOR A
68  190                     CP (HL)
69  32 33                   JR NZ,L6
71  6 5                     LD B,5
73  16 254            L4    DJNZ L 4
75  54 23             L5    LD (HL),23
77  227                     EX (SP),HL
```

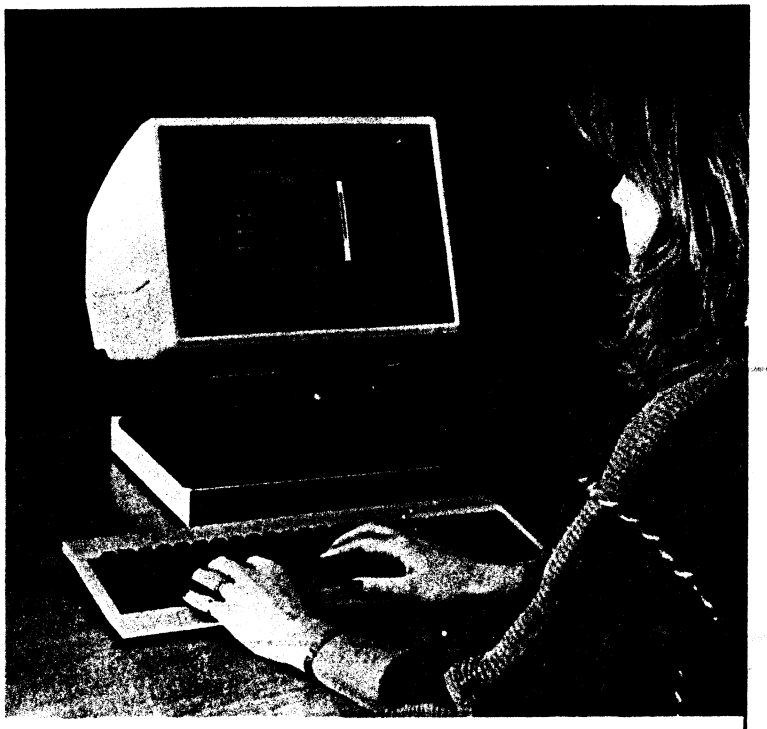
Maskinkoddel för "sifferjakt". Se i RT 1982 nr 11 hur man bäst kan mata in den!

SIFFERJAKT

```
REM        (115 tecken maskinkod)
FOR I=0 TO 10;"xxxxxxxxxxxx"
    I=7 TO 10;"mSIFFERJAKTm"
FOR I=0 TO 10;"xxxxxxxxxxxx"
PRINT AT 6,0;"DATORN PLACERAR SLUMPVIS UT","NIO SIFFROR INOM EN SPELPLAN.",,,
"DU SKA,GENOM ATT STYRA PRICKEN","MED RATTARNA,SAMLA IN ALLA","SIFFRORNA I TUR
OCH ORDNING.",,,,"TIDEN REGISTRERAS UNDER JAKTEN.",,,,,"LYCKA TILL.";AT 21,0;
"(START-TRYCK ""NEWLINE"".)"
PAUSE 4E4
POKE 16389,128
RAND
DIM S$(9)
CLS
PRINT TAB 11;"SIFFERJAKT"
PRINT ,,TAB 7;"XXXXXXXXXXXXXXXXX"
FOR I=3 TO 18
PRINT TAB 7;"X";TAB 24;"X"
NEXT I
    TAB 7;"XXXXXXXXXXXXXXXXX"
   =1 TO 9
  $(I)=CHR$ INT (RND*255)
FOR J=1 TO I-1
IF S$(J)=S$(I) THEN GOTO 170
NEXT J
LET J=INT (CODE S$(I)/16)
PRINT AT 3+J,23-CODE S$(I)+16*J;I
NEXT I
GOTO 280
PRINT "RATTARNA SKALL VARA VRIDNA FULLT MOTURS."
PAUSE 50
PRINT AT 20,0;"          (40 mellanslag)
POKE 2133,0
LET I=PEEK 2183
POKE 21001,0
PRINT AT 20,0;
   >2 OR PEEK 21E3>2 THEN GOTO 250
  J=USR 16515
  I=PEEK 16514
PRINT "OTUR...DU TOG FEL SIFFRA EFTER" AND NOT I;"DU KLARADE DET PAA" AND I,
J/50;" SEKUNDER"
SCROLL
PRINT "VILL DU SPELA IGEN?(J/N)"
PAUSE 4E4
IF INKEY$="J" THEN GOTO 90
CLS
PRINT AT 10,6;"xxxxxxxxxxxxxxxxxxxxxx"
PRINT TAB 6;"m SIFFERJAKTEN SLUT m"
PRINT TAB 6;"xxxxxxxxxxxxxxxxxxxxxx"
```

Glöm inte FAST om programmet skall köras på ZX 81!

Här är basicdelen av det nya "sifferjakt"-programmet. Observera, att man måste ha en a/d-omvandlare ansluten för att programmet skall kunna köras! Hur man bygger en sådan finns beskrivet i RT 1982 nr 11. Till programmet hör också en maskinkoddel.



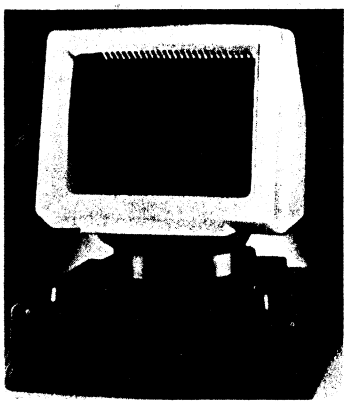
P2500 – Ny dator från Philips

P 2500 heter en ny medlem i P 2000-familjen – en storebror till den tidigare P 2000-versionen från Philips.

Förbättringarna omfattar främst ökad snabbhet, minneskapacitet upp till 320 K samt ergonomiskt lågprofiltangentbord, flexibilitet och utbyggnadsmöjligheter. Datakommunikationen hanteras av slavprocessorer som avlastar huvudprocessorn. Normal bearbetning kan därför ske med bibehållen hastighet.

P 2500 är främst avsedd för yrkesmässig användning. Ett komplett system, exkl skrivare, kostar 32 000 kronor exkl moms. Pris för skrivaren Epson MX 80 5 900 kronor. Avancerade skrivare som TEC, Tally och Fujitsu finns från 9 000 kronor och uppåt.

I samband med introduktionen av P 2500 övergår ansvaret för marknadsföring och försäljning av samtliga mikrodatorer i Philips P 2000-serie till Management Computer International – MCI, *tel 08/23 21 60.*



Nytt hölje för din dator

Peerless Foam Mouldings Co Ltd. presenterar *"Orbitor"*, ett nytt dataterminalsystem i tre delar för elektronikindustrin.

◄ *Mikroprocessorlådan* består av över- och underdel i polystyrol. Front- och bakstycke kan fås i stålplåt eller eloxerad aluminium. De kan lätt modifieras eller bearbetas enligt kundönskemål. Lådan är främst avsedd som interfacelåda och har plats för två 5 1/4″-disc-drives eller ett stort kretskort. Invändiga mått: 445×305×höjd 116 mm, höjden kan ökas med hjälp av sidopaneler.

◄ *Monitorlådan* i Orbitor-serien är avsedd för ett 14″ – alternativt 15″ – färgbildrör med 90° avlänkning.

◄ Den tredje delen i systemet är *sockeln* som är rörlig framåt 5° och bakåt 15° från vertikalplanet och svängbar ±45°. Anslutningar och kontaktdon är åtkomliga i sockelns bas.

Generalagent: **ELFA RADIO & TELEVISION AB,** avd Elektromekaniska byggsystem, tel 08/730 07 00.

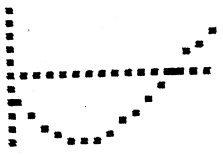
PROGRAMBØRSEN

ZX-biorytmer

Under titlen "Program-Børsen" vil ALT-OM-DATA fremover præsentere sjove og/eller praktiske programmer til markedets populære micro- og hobby-datamater. Alle læsere er velkomne til at deltage i "Program-Børsen", og vi betaler naturligvis fuldt honorar for de godkendte programmer.
Redaktionen stiller nogle enkelte krav til de indsendte dataprogrammer:
1. De skal være udviklet til de mest populære datamater på det danske marked. Det vil sige Sinclair ZX-81, Commodore VIC-20, Texas TI-99, Acorn Atom, ABC 80, eller Nascom. Eventuelt kan andre også komme på tale.
2. Programmerne skal helst være hjemmelavede. De må altså ikke være kopier af de programmer, som firmaerne selv tilbyder. Programmerne må heller ikke have været offentliggjort i andre danske tidsskrifter.

Det følgende biorytmeprogram er udviklet til en ZX-81 med 16K RAM.
Biorytmer — regelmæssige variationer i helbred, psyke og forholdet til andre mennesker — var allerede kendt for over 2000 år siden. Dengang var det den græske læge Hippokrates, som mente, at han havde fundet et system, som kunne fortælle, hvordan et bestemt menneske havde det på en bestemt dag. I dag tilskrives det moderne biorytme system den østrigske læge Dr. Fliess.
Programmet udregner de nøjagtige biorytmer for den ønskede sag og beregner samtidig en gennemsnitskurve for den følgende måned. Som grundlag indtastes fødselsdatoen for den person, man ønsker målingen på.
Når programmet indtastes, starter man med et lille hjælpeprogram, som lægger maskinkoden på plads. Indtast blot de trecifrede tal i den rækkefølge, vi har vist. Efter kørslen vil maskinkoden ligge i linie 0. De øvrige linier slettes ved at indtaste hovedprogrammet ovenpå dem. Vi har indbygget en del REM-sætninger og "Spaces" for at lette forståelsen. De kan dog undværes. For at få mere plads på skærmen, benytter vi et lille trick i POKE sætningen i linie 910, som giver 24 linier på skærmen.

```
BIORYTME FOR:  ALFREDO HITSHOCK
FØDT DEN:      23- 8-1956
BIORYTME DAG:   1-12-1981

FYSIK:          -0.4
PSYKE:          -0.43
INTELLEKT:      -0.46
GENNEMSNIT:     -0.43

PLUS DAGE
```



```
MINUS DAGE
```

```
BIORYTME FOR:  ALFREDO HITSHOCK
FØDT DEN:      23- 8-1956
BIORYTME DAG:   1-12-1981

FYSIK:          -0.4
PSYKE:          -0.43
INTELLEKT:      -0.46
GENNEMSNIT:     -0.43

PLUS DAGE
```

```
MINUS DAGE
```

Eksempler på udskrift

Hovedprogram

```
      Ø  REM ) ?  ??  ?M  RND  ?M  RND
GOSUB PICOS ??  5    ?? GOSUB ?$RND
5     U77 RETURN  4C   ?Q   F?/ PAUSE
/LN
 1Ø   REM  ************************
 2Ø   REM  *                      *
 3Ø   REM  * BIORYTME PROGRAM     *
 4Ø   REM  * COPYRIGHT:           *
 5Ø   REM  * MICHAEL SØRENSEN     *
 6Ø   REM  * NY ELEKTRONIK        *
 7Ø   REM  * FOR ZX81 MED 16K     *
 8Ø   REM  ************************
 9Ø   REM
      INDDATA
1ØØ   PRINT AT Ø , Ø ;
      "BIORYTME FOR: "
11Ø   PRINT AT 2Ø , Ø ;
      "SKRIV DIT NAVN"
12Ø   PRINT AT 21 , Ø ;
      "OG TRYK *NEW LINE*"
13Ø   INPUT A$
14Ø   PRINT AT Ø , 14 ; R$
15Ø   PRINT AT 1 , Ø ; "FØDT DEN:"
16Ø   PRINT AT 2Ø , 6 ;
      "DIN FØDSELSDAG"
17Ø   INPUT FD
18Ø   PRINT AT 1 , 14 + (FD <1Ø) ;
      FD ; "—"
19Ø   PRINT AT 2Ø , 17 ;
      "MAANED"
2ØØ   INPUT FM
21Ø   PRINT AT 1 , 17 + (FM <1Ø) ;
      FM ; "—"
22Ø   PRINT AT 2Ø , 6 ;
      "DIT FØDSELSAAR"
23Ø   INPUT FY
24Ø   PRINT AT 1 , 2Ø ; FY
25Ø   PRINT AT 2 , Ø ;
      "BIORYTME DAG :"
26Ø   PRINT AT 2Ø , 6 ;
      "DIN BIORYTME DAG"
27Ø   INPUT BD
28Ø   PRINT AT 2 , 14 + (BD <1Ø) ;
      BD ; "—"
29Ø   PRINT AT 2Ø , 19 ;
      "MAANED"
3ØØ   INPUT BM
31Ø   PRINT AT 2 , 17 + (BM <1Ø) ;
      BM ; "—"
32Ø   PRINT AT 2Ø , 6 ;
      "DIT BIORYTME AAR"
33Ø   INPUT BY
34Ø   PRINT AT 2 , 2Ø ; BY
35Ø   REM
      PRINT 22 SPACES
36Ø   PRINT AT 2Ø , Ø ;
37Ø   REM
      PRINT 17 SPACES
38Ø   PRINT AT 21 , Ø ;
4ØØ   REM
      HER STARTER UDREGNINGERNE
41Ø   LET SD=Ø
```

```
500  REM
     UDREGN DET RESTERENDE
     ANTAL DAGE I AARET
510  LET D=FD
520  LET M=FM
530  LET Y=FY
540  GOSUB 2000
550  LET AD= -X - ( M >2
     AND Y/4 -INT (Y/4) =0
     AND Y/100 -INT (Y/100) <>0)
600  REM
     UDREGN ANTALLET
     AF SKUDDAGE
610  FOR Z=FY TO BY
620  IF Z/4-INT (Z/4) =0
     AND Z/100-INT (Z/100) <>0
     AND FY <>BY
     THEN LET SD=SD+1
630  NEXT Z
650  REM
     UDREGN ANTALLET AF DAGE
     FRA FØDSLEN TIL BIORYTME
     AARET BEGYNDER
660  LET AD=AD+SD+365* (BY-FY)
700  REM
     UDREGN ANTALLET AF DAGE
     DER ER GAAET I
     BIORYTME AARET
710  LET D=BD
720  LET M=BM
730  LET Y=BY
740  GOSUB 2000
750  LET AD=AD+X
770  REM
     UDREGN BIORYTMEN
780  GOSUB 2500
800  REM
     UDSKRIV BIORYTME DAGEN
810  PRINT AT 2 , 17 ; "  ";
     AT 2 , 17+ (BM <10) ; BM
820  PRINT AT 2 , 20 ; BY
830  REM
     UDSKRIV BIORYTMEN
840  PRINT AT 4 , 0 ; "FYSIK :" ;
     AT 4 , 14 ; "
     AT 4 , 15- (INT (99*A+.5) <0) ;
     INT (100*A+ .5) / 100
850  PRINT AT 5 , 0 ; "PSYKE : " ;
     AT 5 , 14 ; "
     AT 5 , 15- (INT (99*B+ .5) <0) ;
     INT (100*B+ .5) /100
860  PRINT AT 6 , 0 ; "INTELLEKT : " ;
     AT 6 , 14 ; "
     AT 6 , 15- (INT (99*C+ .5) <0) ;
     INT (100*C+ .5) /100
870  PRINT AT 7 , 0 ; "GENNEMSNIT :" ;
     AT 7 , 14 ; "
     AT 7 , 15- (INT (99*D+ .5)·<0) ;
     INT (100*D+ .5) /100 ;
880  PRINT AT 9 , 0 ;
     "PLUS DAGE"
890  PRINT AT 21 , 0 ;
     "MINUS DAGE"
900  REM
     UDSKRIV TEKST PAA
     LINIE 22 OG 23
910  POKE 16418 , 0
920  PRINT AT 22 , 8 ;
     "BIORYTME DAGEN"
930  PRINT AT 23 , 4 ;
     "OG DE FØLGENDE 30 DAGE"
950  REM
     UDSKRIV ET KOORDINATSYSTEM
960  FOR P=2 TO 22 STEP 2
970  PLOT 62 , P
980  NEXT P
1000 REM
     UDSKRIV BIORYTME KURVEN
1010 for G=AD TO AD+30
1020 GOSUB 2520
1030 PLOT 63 , INT (10*D+12.5)
1040 REM
     FLYT KURVEN
1050 RAND USR (16514)
1060 PLOT 62 , 12
1070 NEXT G
1100 REM
     BEREGN NÆSTE MAANED
1110 LET BM=BM+1
1120 IF BM >12 THEN LET BY=BY+1
1130 IF BM >12 THEN LET BM=BM-12
1200 REM
     PAUSE RUTINE
1210 FOR Q=BY-FY TO 250
1220 REM
     HOLD EN LILLE PAUSE
1230 NEXT Q
     BEGYND FORFRA
1310 GOTO 400
2000 REM
     BEREGN ANTALLET AF DAGE
     FRA DEN 1/1
2010 LET X=D + (M >2
     AND Y/4-INT (Y/4)=0
     AND Y/100-INT (Y/100) <>0)
2020 IF M= 2 THEN LET X= 31+X
2030 IF M= 3 THEN LET X= 59+X
2040 IF M= 4 THEN LET X= 90+X
2050 IF M= 5 THEN LET X=120+X
2060 IF M= 6 THEN LET X=151+X
2070 IF M= 7 THEN LET X=181+X
2080 IF M= 8 THEN LET X=213+X
2090 IF M= 9 THEN LET X=242+X
2100 IF M=10 THEN LET X=273+X
2110 IF M=11 THEN LET X=303+X
2120 IF M=12 THEN LET X=334+X
2130 RETURN
2500 REM
     UDREGN BIORYTMEN
2510 LET G=AD
2520 LET A=SIN (2*PI*G/23)
2530 LET B=SIN (2*PI*G/28)
2550 LET C=SIN (2*PI*G/33)
2550 REM
     UDREGN GENNEMSNITTET
2560 LET D= (A+B+C) /3
2570 RETURN
9700 REM
     SÆT BAANDOPTAGEREN
     TIL OPTAGELSE OG
     TRYK RUN 9800
9800 SAVE "BIORYTME"
0000 RUN
```

```
  5   REM (54 TEGN) 34567890123
 4567890123456789012345678901234
 10   REM
      HJÆLPE-PROGRAM
 20   FOR A=16514 TO 16567
 30   INPUT B
 40   PRINT TAB 4*(A-16514),B;
 50   POKE A, B
 60   NEXT A
 70   POKE 16510 , 0
```

16514	17	16532	237	16550	9
16515	74	16533	66	16551	175
16516	1	16534	200	16552	60
16517	98	16535	98	16553	35
16518	107	16536	107	16554	35
16519	175	16547	1	16555	254
16520	125	16538	33	16556	32
16521	50	16539	0	16557	40
16533	164	16540	9	16558	7
16523	64	16541	84	16559	70
16524	175	16542	93	16560	54
16525	124	16543	237	16561	0
16526	50	16544	75	16562	43
16527	165	16545	12	16563	112
16428	64	16546	64	16564	24
16529	1	16547	33	16565	242
16530	181	16548	181	16566	24
16531	2	16549	2	16567	205

Maskinkode rutine

PC 1211 'kapital'

■ Økonomi beskæftiger alle mennesker på den ene eller anden måde. Det gælder naturligvis også for de forskellige opsparingsformer, som gerne skulle formere formuen på kortest mulig tid.
Ved hjælp af det følgende program kan man på en snild måde finde ud af, hvor meget man får ud af sin opsparing. Programmet er skrevet på Sharps lille PC-1211 lommedatamat, men skulle også med nogle modificeringer kunne overføres til andre typer computere.
Lad os sige, at vi sætter 850 kr. i banken, og at denne giver 12% i rente. Hvor meget har vi så efter 5 år? Det udregnes således (beløb i kr.):

$$850+850\cdot\frac{12}{100}=952$$

$$952+952\cdot\frac{12}{100}=1066$$

$$1066+1066\cdot\frac{12}{100}=1194$$

$$1194+1194\cdot\frac{12}{100}=1337$$

$$1337+1337\cdot\frac{12}{100}=1498$$

Udregningen kan også skrives lidt kortere:

$$850\cdot(1+\frac{12}{100})=952$$

$$952\cdot(1+\frac{12}{100})=1066$$

$$1066\cdot(1+\frac{12}{100})=1194$$

$$1194\cdot(1+\frac{12}{100})=1337$$

$$1337\cdot(1+\frac{12}{100})=1498$$

Vi bliver her ved med at gange med den samme parentes

$$(1+\frac{12}{100}).$$

Hvis renten havde været R%, skulle parentesen være

$$(1+\frac{R}{100}).$$

Program

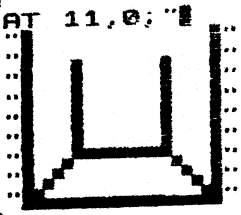
Programmet laves på lommedatamaten PC 1211. Første udkast til det kan se således ud:

```
60: INPUT "START
    KAPITAL? ";K
70: INPUT "RENTE
    FOD (%)? ";R
80: INPUT "ANTAL
    TERMINER? "
    ;N
90: LET R=R/100
100: PRINT " "
110: PRINT " "
120: PRINT "ANTAL
    "
130: PRINT "TER-
        MI-"
140: PRINT "MINER
        PITAL"
150: PRINT " "
160: PRINT 0;K
170: FOR I=1TO N
180: LET K=K*(1+R
    )
190: PRINT I;K
200: NEXT I
210: PRINT " "
```

I programmet benytter vi samme fremgangsmåde som ovenfor beskrevet. De første linjer (60, 70 og 80) går til at indlæse den indsatte kapital (startkapitalen), renten pr. termin (rentefoden) og antallet af forrentningsperioder (terminer).
I linje 90 omregnes rentefoden fra procent til absolut værdi. Dernæst springes et par linjer frem på printeren (skriveren) ved hjælp af linje 100 og 110. Det er kun for skønheds skyld. Så laves der overskrift i linje 120, 130 og 140 på printeren, og igen en blank linje for et syns skyld. I sætning 160 udskrives 0 (antallet af terminer ved starten) og startkapitalen.

LABYRINT MED 3-D UDSIGT

```
    1 REM 13-8-82
    2 REM (C) THOMAS CORELL
    3 REM OG  KASPER VAD.
    4 CLS
    5 PRINT "DO YOU WANT A RADAR?
"
    6 INPUT R$
    7 IF (R$(1)<>"N")*(R$(1)<>"Y"
)=0 THEN GOTO 3
    8 CLS
    9 LET MOV=0
   10 FAST
   20 DIM A(225)
   30 FOR N=1 TO 15
   40 LET A(N)=1
   50 LET A(N+210)=1
   60 LET A(N*15)=1
   70 LET A(N*15-14)=1
   80 NEXT N
   90 FOR N=2 TO 12 STEP 2
  100 LET X=INT (RND*11)+1
  110 FOR M=0 TO 3
  120 LET A(15*N+X+M)=1
  130 NEXT M
  140 NEXT N
  150 FOR N=2 TO 11 STEP 2
  160 LET X=INT (RND*10+2)
  170 FOR M=0 TO 45 STEP 15
  180 LET A(15*X+M+N)=1
  190 NEXT M
  200 NEXT N
  210 PRINT AT 4,0;"MOVE: N=NORTH
                       E=EAST,
 S=SOUTH,
 W=WEST."
  220 PRINT AT 7,0;" TO GIVE UP:
G"
  221 PRINT AT 9,0;" TO TURN 180
DEG: T"
  222 PRINT AT 2,0;"MOVES:"
  230 LET A(203)=0
  235 LET A(8)=0
  240 LET P=203
  245 LET LM=-15
  250 IF R$(1)="N" THEN GOTO 256
  251 FOR N=1 TO 15
  252 FOR M=48 TO 63
  253 PLOT M,N
  254 NEXT M
  255 NEXT N
  256 SLOW
  257 PRINT AT 0,0;"YOU ARE LOOKI
NG"
  258 IF R$(1)="Y" THEN PRINT AT
12,26;"N"
  259 GOTO 260
  260 GOSUB 9890
  261 LET A(P)=2
  265 UNPLOT 55,15
  270 IF LM=-15 THEN GOSUB 9750
  280 IF LM=1 THEN GOSUB 9700
  290 IF LM=15 THEN GOSUB 9650
  300 IF LM=-1 THEN GOSUB 9600
  305 PRINT AT 2,8;MOV
  310 LET Y=INT (P/15)
  320 LET X=P-Y*15
  325 LET Y=15-Y
  330 UNPLOT 47+X,Y
  340 INPUT Z$
  342 IF Z$="T" THEN LET LM=-LM
  342 IF Z$="T" THEN GOTO 260
  345 IF Z$="G" THEN GOTO 500
  346 IF Z$="N" THEN LET LM=-15
  360 IF Z$="S" THEN LET LM=15
  370 IF Z$="E" THEN LET LM=1
  380 IF Z$="W" THEN LET LM=-1
  390 IF Z$="W" THEN LET LM=-1
  400 IF A(P+LM)=1 THEN GOTO 340
  410 LET P=P+LM
  415 IF R$(1)="Y" THEN PLOT 47+X
,Y
  417 LET MOV=MOV+1
  420 IF P<>8 THEN GOTO 260
  425 CLS
  430 PRINT AT 0,0;"YOU HAVE ESCA
PED THE MAZE."
  500 PRINT AT 1,0;"IN ";MOV;" MO
VES"
  505 FOR M=0 TO 210 STEP 15
  510 FOR N=1 TO 15
  520 IF A(N+M)=1 THEN PRINT "■";
  525 IF A(N+M)=2 THEN PRINT "■";
  530 IF A(N+M)=0 THEN PRINT " ";
  540 NEXT N
  550 PRINT
  560 NEXT M
  570 PRINT "CARE FOR ANOTHER GAM
E?"
  580 INPUT Z$
  590 IF Z$(1)="Y" THEN RUN
  600 IF Z$(1)<>"N" THEN GOTO 580
  610 STOP
  620 SAVE "3D■"
  630 RUN
 9600 IF A(P-1)=1 THEN GOSUB 9790
 9610 IF A(P-15)=1 THEN GOSUB 981
0
 9620 IF A(P+15)=1 THEN GOSUB 985
0
 9630 PRINT AT 0,16;"W"
 9640 RETURN
 9650 IF A(P+15)=1 THEN GOSUB 979
0
 9660 IF A(P-1)=1 THEN GOSUB 9810
 9670 IF A(P+1)=1 THEN GOSUB 9850
 9680 PRINT AT 0,16;"S"
 9690 RETURN
 9700 IF A(P+1)=1 THEN GOSUB 9790
 9710 IF A(P+15)=1 THEN GOSUB 981
0
 9720 IF A(P-15)=1 THEN GOSUB 985
0
 9730 PRINT AT 0,16;"E"
 9740 RETURN
 9750 IF A(P-15)=1 THEN GOSUB 979
0
 9760 IF A(P+1)=1 THEN GOSUB 9810
 9770 IF A(P-1)=1 THEN GOSUB 9850
 9780 PRINT AT 0,16;"N"
 9785 RETURN
 9790 PRINT AT 13,3;"■■■■■"
 9800 RETURN
 9810 PRINT AT 11,9;"■"
 9820 PRINT AT 12,8;"■"
 9830 PRINT AT 13,7;"■"
 9840 RETURN
 9850 PRINT AT 11,0;"■"
 9860 PRINT AT 12,1;"■"
 9870 PRINT AT 13,2;"■"
 9880 RETURN
 9890 PRINT AT 11,0;"■                ■"
 9900 PRINT "                  "
 9910 PRINT "                  "
 9920 PRINT "                  "
 9930 PRINT "                  "
 9940 PRINT "                  "
 9945 PRINT "                  "
 9950 PRINT "                  "
 9960 PRINT "                  "
 9970 PRINT "                  "
 9980 RETURN
```



3-D labyrint

Dette sjove program kræver en 16K RAM udvidelse til ZX-81. Datamaten laver en ny labyrint for hvert spil, således at man aldrig kommer ud på samme opgave to gange. Opgaven gælder i øvrigt, at man kommer ud af labyrinten på så få træk som muligt.

Når programmet er indtastet, og man starter ved at indkode RUN + NEWLINE, spørger datamaten, om man ønsker en radar. Ved tryk på Y siges ja, ved tryk på N nej.

Maskinen udtænker derefter en labyrint og spørger i hvilken retning, man ønsker at gå. N indtastes derpå for nord, S for syd, E for øst og W for vest. Hvis man opgiver undervejs, trykkes G.

Efter hvert træk vises et 3-dimensionelt billede af de retninger, man kan gå i. Hvis man ønsker at se det område, man netop har passeret, tastes T, hvorefter hele billedet drejer 180 grader.

Efter endt spil vises labyrinten i fuld størrelse sammen med den rute, man har valgt mellem forhindringerne.□

Räkneknep för basic-grafik

★ *Den här artikeln skall vi helt ägna åt grafik på ZX81, och främst rörlig grafik. En mängd program för nöjsam förströelse bygger på tekniken att kunna åstadkomma rörliga bilder.*

★ *Artikeln bygger på boken "Mer om BASIC ZX81", Studieförlaget, Uppsala.*

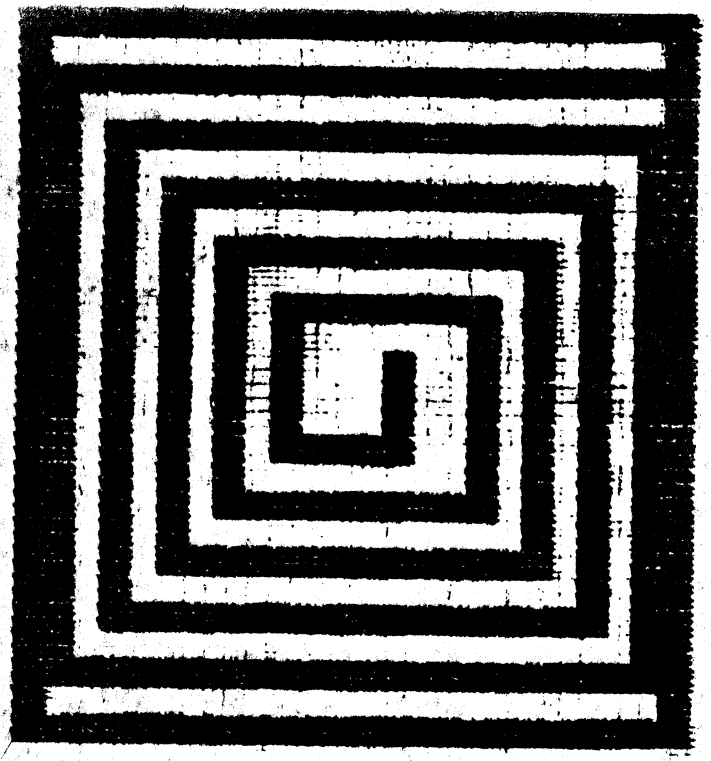


Fig 1. Exempel på en manuellt ritad grafikbild på ZX81.

Även om datorn ZX81 mer än väl har de språkliga resurserna för konstruktion av rörlig grafik har den några andra begränsningar som måste beaktas. En sådan är upplösningen i bara 64×44 punkter. Det gör att vi inte kan åstadkomma hur fina figurer eller noggranna kurvor som helst. Ett annat, och kanske mer begränsande problem, är den speciella tekniken för bildalstringen, som gör grafiken långsam. Basicspråket är i sig inget snabbt språk och tekniken i ZX81 gör inte saken bättre. Det är därför viktigt att känna till och kunna utnyttja många knep som möjligt för att få fart på bilderna. För riktigt snabb grafik krävs kunskap i maskinkodsprogrammering, men det är en helt annan historia. Nu gäller det basic, och mycket roligt kan åstadkommas även med enkla medel.

Styr rörelsen med variabler

En PLOT-punkt (eller PRINT-symbol) kan flyttas i åtta riktningar. Se fig 1! Rörelseriktningen bestäms enkelt med två *riktningsvariabler*:

H för horisontell riktning
V för vertikal riktning

H och V får bara anta värdena 0, -1 och 1 om vi önskar en så sammanhängande och ryckfri rörelse som möjligt. Från en godtycklig position X, Y bestämmer vi nästa position med riktningsvariablerna:

X + H, Y + V

Genom att arbeta med riktningsvariabler kan vi skapa en "standard" för grafikrutiner, som kan användas för en rad olika rörelsemönster och dessutom ofta leder till korta och därmed snabba rutiner.

Standardrutin för rörlig grafik

Standardrutinen för rörlig grafik har sex byggbitar:

1. Bestäm startpunkt (X och Y).
2. Rita i X, Y.
3. Bestäm riktningsvariablerna (H och V).
4. Bestäm nästa position (X + H och Y + V).
5. Ta bort punkten från förra positionen.
6. Ta om från 2.

Genom att utgå från standardrutinen får vi snabbt en fungerande rutin som sedan kan finputsas och förenklas för bästa (= snabbaste) exekvering. Program 1 är ett enkelt exempel på rutinen för en diagonal rörelse från startpunkten 0,0.

Manuellt styrd rörelse

Ofta vill man styra ett rörelseförlopp från tangentbordet. För att åstadkomma detta används funktionen

INKEY$

som läser in ett godtyckligt tecken från tangentbordet. Tecknet tas in i form av en sträng. Vi väljer att styra rörelsen med tangenterna:

8 för högerriktning
5 för vänsterriktning
7 för riktning uppåt
6 för riktning nedåt

När vi använder oss av standardrutinen behöver vi bara formulera lämpliga satser för bestämning av riktningsvariablerna H och V. Närmast till hands ligger kanske att använda IF . . . THEN-satser:

```
300 IF INKEY$ = "8" THEN LET H = 1
310 IF INKEY$ = "5" THEN LET H = -1
320 IF INKEY$ = "7" THEN LET V = 1
330 IF INKEY$ = "6" THEN LET V = -1
340 IF INKEY$ = "" THEN LET H = 0
350 IF INKEY$ = "" THEN LET V = 0
```

Nu finns en bättre lösning som utför samma arbete på bara två programrader.

Studera det *logiska* uttrycket

INKEY$ = "8"

Om uttrycket är sant (övi vi trycker på tangenten 8) är parentesens värde = 1. Om osant är värdet = 0. Genom att kombinera de logiska uttryck som är aktuella får vi H och V ur raderna:

```
300 LET H = (INKEY$ = "8") - (INKEY$ = "5")
310 LET V = (INKEY$ = "7") - (INKEY$ = "6")
```

Den färdiga rutinen för manuell styrning är listad i program 2.

En rolig variant av rutinen får vi genom att ta bort rad 500. Ett slags *Etch a Scetch* för att rita bilder. Med COPY-instruktionen kan bilden sparas med skrivare (fig 1).

Slumpmässigt styrd rörelse

En andra variant av rörelse är den som styrs av "slumpen". Genom att byta ut raderna 300 och 310 mot:

```
300 LET H = (INT (RND + .5) = 1) - (INT (RND + .5) = 0)
310 LET V = (INT (RND + .5) = 1) - (INT (RND + .5) = 0)
```

I vår standardrutin får vi slumpmässig rörelse i fyra riktningar. Se program 3!

Begränsa bildutrymmet

När vi försöker "tända" ett bildelement utanför de tillåtna koordinatgränserna får vi antingen programavbrott eller teckenbyte (vid försök att passera X/Y-axlarna) på riktningsvariablerna. Detta kan undvikas genom att man begränsar rörelseytan så att koordinatgränserna aldrig kan passeras. Också detta kan göras med logiska uttryck, och efter lite tankearbete direkt vid bestämningen av H och V. Vi skall ge ett par exempel.

I rutinen för manuell styrning ändrar vi raderna 300 och 310:

```
300 LET H = (INKEY$ = "8") - (INKEY$ = "5") + (X<1) - (X>62)
310 LET V = (INKEY$ = "7") - (INKEY$ = "6") + (Y<1) - (Y>42)
```

Av Thomas Eriksson

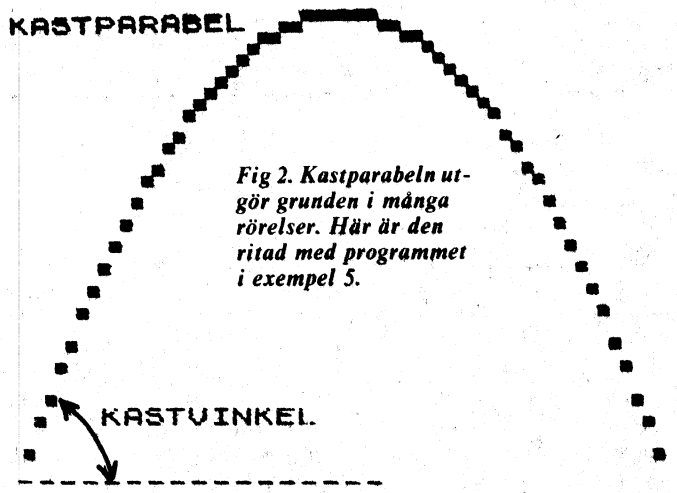


Fig 2. Kastparabeln utgör grunden i många rörelser. Här är den ritad med programmet i exempel 5.

Program 1. "Standardrutin" för rörlig grafik.

```
 10 REM STANDARDRUTIN
100 LET X=0
110 LET Y=0
200 PLOT X,Y
300 LET H=1
310 LET V=1
400 LET X=X+H
410 LET Y=Y+V
500 UNPLOT X-H,Y-V
600 GOTO 200
```

Hur vi än anstränger oss kommer gränserna att vara omutliga. Vad är det då vi gjort? Ja se på rad **300**! Så länge vi håller **8** nedtryckt kommer första parentesen (**INKEY$ = "8"**) att ha värdet 1. Alla övriga parenteser är = 0 ända till dess vi får ett X-värde som överstiger 62! Då blir sista parentesen (**X>62**) sann (=1) och minustecknet framför ser till att hela uttrycket, dvs **H**-värdet, blir 0. X kan med andra ord aldrig bli större än 62! På samma sätt fungerar uttrycken för de andra gränserna.

Studsande boll

Tekniken för att begränsa rörelseytan kan vi använda för att kapa en "boll" som studsar mot koordinatgränserna. Vi låter bollen starta mitt på skärmen och röra sig diagonalt upp åt höger (**H** och **V** = 1 i utgångsläget). Uttrycken för **H** och **V** blir:

H = (X=0) − (X=63) + H* (X>0 AND X<63)

V = (Y=0) − (Y=43) + H* (Y>0 AND Y<43)

Vi avstår från att förklara uttrycken närmare. Sätt in dem i standardrutinen och prova resultatet. Se program 4! Du måste ha 16 k RAM för att köra programmet. Har du inte tillgång till extra minne så minska bildytan genom att ändra övre gränsvärdet för X-variabeln till 40. Uttrycket på rad **300** blir då:

H = (X=0) − (X=40) + H* (X>0 AND X <40)

Kaströrelse på skärmen

Kaströrelsen eller kastparabeln är användbar, inte minst i spelprogram, för simulering av kast och liknande. Här är kastparabeln intressant eftersom den representerar ett vanligt problem: Hur man grafiskt åskådliggör en matematisk funktion. Vi börjar med att studera funktionen.

Kastparabeln följer sambandet:

$y = Ax - Bx^2$

där **A** och **B** beror på kastvinkeln och projektilens utgångshastighet. Tar vi med också dessa variabler blir sambandet:

$y = (\tan a)\,x - (9.81/(2v^2\cos^2 a))x^2$

där **a** är kastvinkeln (se fig 2) och **v** utgångshastigheten. Med **a** och **v** givna får kurvan ett bestämt utseende. Ändrar vi **a** och/eller **v** får kurvan ett annat förlopp. Genom att skriva ett litet program som ritar funktionen och där vi kan laborera med olika **a**- och **v**-värden kan vi studera detta. För att göra det enklare använder vi det första sambandet ovan. Omskrivet för datorn blir uttrycket:

Y = A*X − B*X*X

Program 5 baseras på en förenklad variant av vår "standardrutin". Rad **420** behövs för att inte kurvan skall "studsa" i X-axeln. Pröva med små ändringar av A- och B-värdena.

Programexemplet 6 utnyttjar sambandet **L = A/B** för att rita parabeln till önskad "nedslagsplats" på X-axeln. ■

Program 2. Utbyggd version av program 1 med manuell styrning av bildpunkten.

```
 10 REM MANUELL STYRNING
100 LET X=30
110 LET Y=20
200 PLOT X,Y
300 LET H=(INKEY$="8")-(INKEY$=
"5")
310 LET V=(INKEY$="7")-(INKEY$=
"5")
400 LET X=X+H
410 LET Y=Y+V
500 UNPLOT X-H,Y-V
600 GOTO 200
```

Program 3. Genom att använda RND-funktionen kan man få slumpmässig styrning av bilden.

```
 10 REM SLUMPM. STYRNING
100 LET X=30
110 LET Y=20
200 PLOT X,Y
300 LET H=(INT (RND+.5)=1)-(INT
(RND+.5)=0)
310 LET V=(INT (RND+.5)=1)-(INT
(RND+.5)=0)
400 LET X=X+H
410 LET Y=Y+V
500 UNPLOT X-H,Y-V
600 GOTO 200
```

Program 4. Om man lägger in gränsvärden kan man få en "boll" att studsa mot "väggarna" runt bilden.

```
 10 REM STUDSANDE BOLL
100 LET X=30
110 LET Y=20
120 LET H=1
130 LET V=1
200 PLOT X,Y
300 LET H=(X=0)-(X=63)+H*(X>0 A
ND X<63)
310 LET V=(Y=0)-(Y=43)+V*(Y>0 A
ND Y<43)
400 LET X=X+H
410 LET Y=Y+V
500 UNPLOT X-H,Y-V
600 GOTO 200
```

Program 5. Program för kastparabeln i fig 2.

```
 10 REM KASTPARABEL
100 LET X=0
110 LET Y=0
200 PLOT X,Y
400 LET X=X+1
410 LET Y=2.5*X-.048*X*X
420 IF Y<=0 THEN STOP
500 GOTO 200
```

Program 6. Utbyggt program som tillåter styrning av kastparabelns längd.

```
 10 REM KASTPARABEL/LAENGD
100 LET X=0
110 LET Y=0
120 PRINT AT 21,1;"ANGE LAENGD
 TILL 60"
130 INPUT L
140 LET A=L*.048
150 CLS
200 PLOT X,Y
400 LET X=X+1
410 LET Y=A*X-.048*X*X
420 IF Y<=0 THEN GOTO 100
500 GOTO 200
```

Minnes schema för ZX81