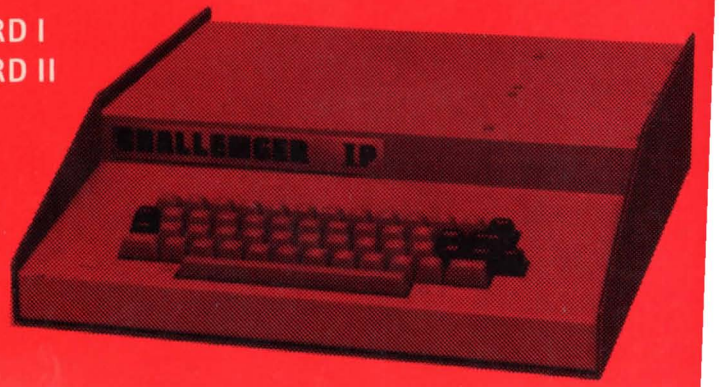
S. Roberts

# The Fifth Book OF OHIO SCIENTIFIC



IMPORTANT PROGRAMS FOR THE

- OSI SUPERBOARD I
- OSI SUPERBOARD II
- C1P, C1PMF
- C4P, C4PMF



ISBN 3–911682–83–5


First Edition
First Printing
August 1982 in the Federal Republic of Germany

Cover Design: W. Hofacker

US-Distributor:
ELCOMP Publishing, Inc.
Postbox 1194
Pomona, California 91769
Tel.: (714) 623-8314

Publisher:
Ing. W. HOFACKER GmbH,
Tegernseerstr. 18, D-8150 Holzkirchen, W.-Germany

# The Fifth Book

## OF

## OHIO SCIENTIFIC

## Table of Content

# Debugger for superboard

DEBUGGER FOR SUPERBOARD

If you have already written programs in machine-language, you probably know how difficult it sometimes is to detect mistakes in such programs.
The definition of breakpoints sometimes helps localize the problem. However it is much better if you can go through a program in single steps. This (and more) can be done by the debugger described in this chapter.
After input and start of the program your screen should look like figure 1.

The debugger waits for the input of the start-address of the program to be tested. The cursor always shows you what you have to enter next.

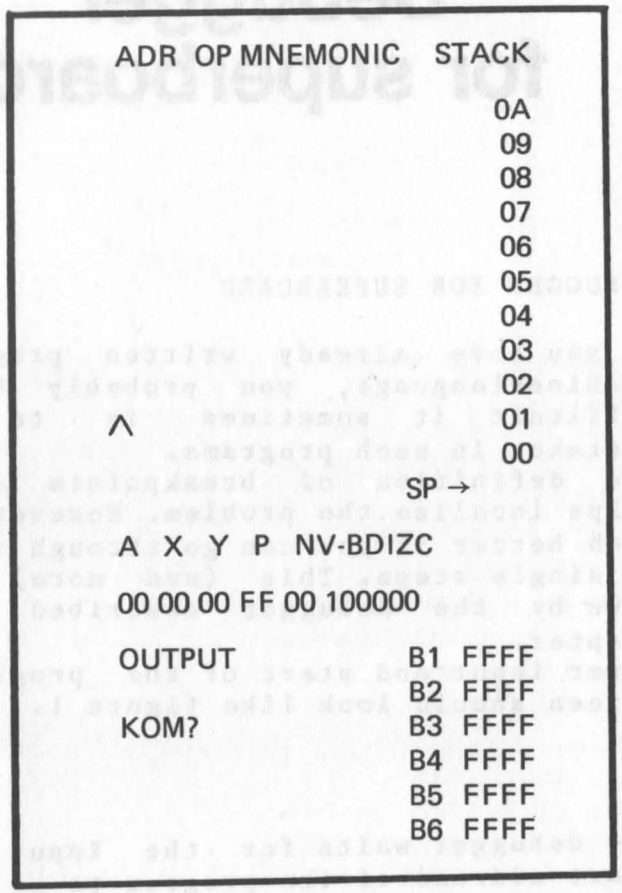



FIGURE 1



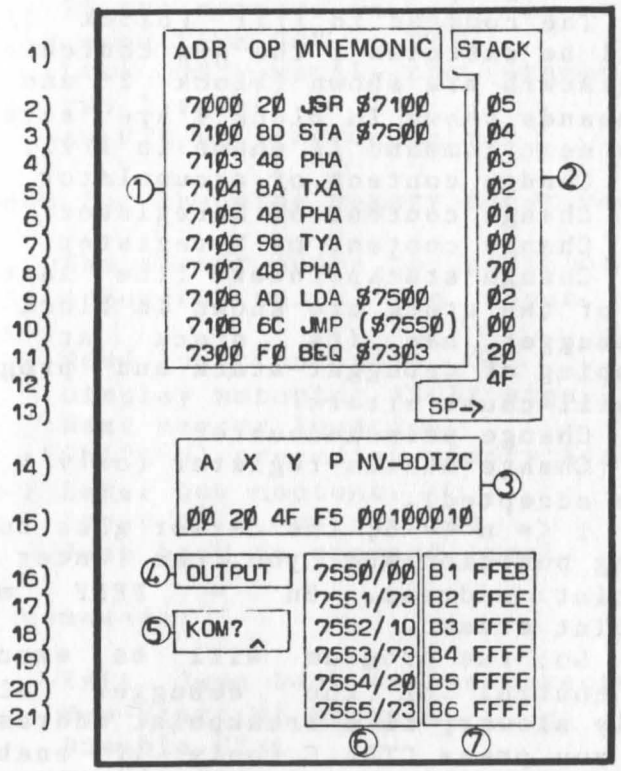



FIGURE 2

When the cursor is beside "KOM?" you have the following options:

BLANK The command in 1/11 (block 1/line 11) will be executed ; the new contents of the registers are shown (block 2 and 3); the commands shown in block 1 are scrolled up; the next command is shown in 1/11.

CA Change content of accumulator

CX Change content of X-register

CY Change content of Y-register

CP Change stackpointer (the last 11 values of the stock are shown in block 2). The debugger has its stack at $28. Overlapping of debugger-stack and program-stack will cause errors!

CC Change programcounter

CS Change status-register (only 1 or 0 will be accepted).

Bn 1 <= n <= 6; the cursor goes to the matching position and you can enter the breakpoint address. Bn = FFFF means breakpoint erased.

G Go; the program will be executed under control of the debugger albeit slightly slower; if a breakpoint addres is hit or you press CTRL C (only if enabled) the program stops and the next command will be shown in 1/11.

Example: You want to test a program beginning at $1000 that works well to $1200. Set the programcounter using the CC command to $1000. Stepping through the program by using the space key up to $1200 would take a very long time. It is easier to define a breakpoint at $1200 and then enter the "G"-command.

K This command is usefull if you want to jump (not execute) one instruction.



Commands for subroutines:

W If the command in 1/11 is a JSR, then the same than "G" to RTS.

S Like "G" until the stackpointer becomes two larger.

R An RTS will be executed.

Commands for changing memory locations:

M The cursor jumps to position 6/21 and the debugger waits for input of an address.

/ Read location again.

" Display matching ASCII sign.

+ Next memory location.

^ (Shift N) preceding memory location.

0-9,A-F Enter new content.

CR Input of a new address.

else Jump back to command level.

Other commands:

E Exit. Jump back to reset-vector.

N New-start of program.

0 Disable CTRL

1 Enable CTRL

CTRL A Show command being executed.

CTRL C Like CTRL A then break.

All commands are echoed in block 5. Invalid commands cause a question mark as an echo. The commands CTRL A and CTRL C only have effect if CTRL has been enabled previously. If you try to go through the routine which reads the keyboard in single steps, it will not work correctly.

Special features of the debugger:

The debugger uses no zero-page address. The output-vector is changed by the debugger so that the characters to be displayed appear

beside the text OUTPUT. This prevents the screen from being overwritten.
If the debugger finds a non-existant OP-code, the address and the contents are displayed and the program branches to the "CC" command.
The command JMP ($XXFF) (where XX = any byte), which is handled wrong by the CPU, is handled right by the debugger.
Another CPU-fault: the PHP command always puts the I-flag = 1 to the stack. For that reason the I-flag always, even after BRK, is given out = 0.

The hex-dump of the debugger.

```
:D1790,1AFF
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
1790  4C 3F 18 00 00 00 FF 20 FF FF FF FF FF FF FF FF
17A0  FF FF FF FF 00 24 24 24 24 24 24 24 24 24 24 24
17B0  24 24 24 24 24 65 D0 46 D2 F6 D2 16 D3 36 D3 56
17C0  D3 76 D3 96 D3 16 D2 25 D3 E5 D2 C5 D1 91 D2 F9
17D0  D2 19 D3 39 D3 59 D3 79 D3 99 D3 2A D3 EC D2 85
17E0  D2 88 D2 8B D2 8E D2 BA D0 8E D3 93 D3 41 44 52
17F0  20 20 4F 50 20 4D 4E 45 4D 4F 4E 49 43 20 20 20
1800  53 54 41 43 4B 00 41 20 20 58 20 20 59 20 20 50
1810  20 4E 56 2D 42 44 49 5A 43 00 42 31 00 42 32 00
1820  42 33 00 42 34 00 42 35 00 42 36 00 53 50 2D 3E
1830  00 4B 4F 4D 3F 00 4F 55 54 50 55 54 00 22 1E D8
1840  A2 28 9A 2C A4 17 30 0F CE A4 17 A2 00 8E AF 17
1850  8A 9D 00 01 E8 D0 F9 20 4F 1D A9 20 A2 00 9D 00
1860  D0 9D 00 D1 9D 00 D2 9D 00 D3 E8 D0 F1 20 F8 1D
1870  CA A0 FF 20 F2 1D C8 B9 ED 17 F0 05 20 08 1E D0
1880  F5 E8 E0 16 D0 ED A0 00 A2 1A 20 F2 1D B9 9E 17
1890  20 A7 1D B9 98 17 20 A7 1D E8 C8 C0 06 D0 EB AD
18A0  3D 18 8D 1A 02 AD 3E 18 8D 1B 02 20 61 1D 4C 50
18B0  19 A2 26 20 F2 1D 20 EB FF A2 20 8E 2B D3 8E 2C
18C0  D3 20 08 1E C9 42 D0 2C 20 EB FF 20 08 1E C9 31
18D0  90 04 C9 37 90 03 4C AE 1A 29 07 E9 00 A8 0A 69
18E0  1A AA 20 F2 1D 20 67 1D 99 9E 17 20 67 1D 99 98
18F0  17 4C B1 18 C9 43 F0 03 4C 77 19 20 EB FF 20 08
1900  1E C9 41 D0 0E A2 2A 20 F2 1D 20 67 1D 8D 93 17
1910  4C B1 18 C9 58 D0 0E A2 2C 20 F2 1D 20 67 1D 8D
1920  94 17 4C B1 18 C9 59 D0 0E A2 2E 20 F2 1D 20 67
1930  1D 8D 95 17 4C B1 18 C9 50 D0 11 A2 30 20 F2 1D
1940  20 67 1D 8D 96 17 20 CB 1C 4C B1 18 C9 43 D0 1A
1950  20 FF 1C A2 16 20 F2 1D 20 67 1D 8D 28 1E 20 67
1960  1D 8D 27 1E 20 3D 1E 4C B1 18 C9 53 F0 03 4C AE
1970  1A 20 75 1C 4C B1 18 C9 20 D0 03 4C B6 1A C9 4E
1980  D0 03 4C 3F 18 C9 45 D0 03 6C FC FF C9 57 D0 0C
1990  AD 96 17 8D B4 17 EE AC 17 4C B6 1A C9 47 D0 06
19A0  EE AD 17 4C B6 1A C9 53 D0 0E AE 96 17 E8 E8 8E
```



```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
19B0  B4 17 EE AC 17 4C B6 1A C9 52 D0 03 4C 51 1B C9
19C0  4B D0 03 4C C4 1B C9 30 D0 08 A9 00 8D AF 17 4C
19D0  B1 18 C9 31 D0 08 A9 FF 8D AF 17 4C B1 18 C9 4D
19E0  F0 03 4C AE 1A 20 12 1D A2 34 20 F2 1D 20 67 1D
19F0  8D 9C 1A 20 67 1D 8D 9B 1A A9 2F 20 08 1E 20 9A
1A00  1A 20 A7 1D 20 EB FF 48 A2 36 20 F2 1D 68 C9 0D
1A10  D0 06 20 C1 1D 4C E5 19 C9 2F D0 08 A2 36 20 F2
1A20  1D 4C FE 19 C9 2B D0 1F EE 9B 1A D0 03 EE 9C 1A
1A30  20 12 1D A2 34 20 F2 1D AD 9C 1A 20 A7 1D AD 9B
1A40  1A 20 A7 1D 4C F9 19 C9 5E D0 0E AC 9B 1A D0 03
1A50  CE 9C 1A CE 9B 1A 4C 30 1A C9 22 D0 0C 20 08 1E
1A60  20 9A 1A 20 08 1E 4C 04 1A 48 A2 36 20 F2 1D 68
1A70  C9 30 30 23 C9 3A 90 08 C9 47 B0 1B C9 41 90 17
1A80  20 08 1E C9 41 90 02 E9 07 29 7F 20 70 1D 20 6D
1A90  1D 20 9E 1A 4C 04 1A 4C B1 18 AD FF FF 60 AE 9C
1AA0  1A AC 9B 1A 8E AC 1A 8C AB 1A 8D FF FF 60 A9 3F
1AB0  20 08 1E 4C B1 18 A0 02 A9 EA 99 91 1B 88 D0 FA
1AC0  20 26 1E 8D A5 17 AD A9 17 D0 09 20 4F 1D 20 5B
1AD0  1D 4C 50 19 C9 0D D0 0E AD A5 17 8D 91 1B A9 04
1AE0  8D 92 1B 4C 7F 1B C9 0B D0 18 A0 02 20 26 1E 99
1AF0  F6 1A 88 D0 F7 C8 B9 FF FF 99 27 1E 88 10 F7 4C
1B00  DD 1B AD A5 17 C9 4C D0 13 A0 02 20 26 1E AA 88
1B10  20 26 1E 8D 27 1E 8E 28 1E 4C DD 1B C9 20 D0 09
1B20  20 4B 1C 8E 96 17 4C 09 1B C9 00 D0 20 20 4B 1C
1B30  AD 97 17 9D 00 01 CA 8E 96 17 09 10 8D 97 17 A0
1B40  01 B9 FE FF 99 27 1E 88 10 F7 4C DD 1B C9 60 D0
1B50  09 AE 96 17 20 63 1C 4C D5 1B C9 40 D0 10 AE 96
1B60  17 E8 BD 00 01 8D 97 17 20 63 1C 4C DD 1B 8D 91
1B70  1B AC B1 17 F0 09 20 26 1E 99 91 1B 88 D0 F7 AE
1B80  96 17 9A AD 97 17 48 AD 93 17 AE 94 17 AC 95 17
1B90  28 EA EA EA 4C AF 1B D8 A0 01 20 26 1E 38 20 13
1BA0  1F AA E8 D0 01 C8 8E 27 1E 8C 28 1E 4C DD 1B 08
1BB0  8D 93 17 8E 94 17 8C 95 17 D8 68 29 EF 8D 97 17
1BC0  BA 8E 96 17 AD B1 17 F0 0C 18 6D 27 1E 8D 27 1E
1BD0  90 03 EE 28 1E EE 27 1E D0 03 EE 28 1E A2 28 9A
1BE0  2C AD 17 30 50 A0 05 B9 98 17 CD 27 1E D0 0E B9
1BF0  9E 17 CD 28 1E D0 06 CE AD 17 4C 45 1C 88 10 E7
1C00  2C AF 17 10 2A A9 FE 8D 00 DF 2C 00 DF 70 20 A9
1C10  FB 8D 00 DF 2C 00 DF 70 06 20 4F 1D 4C 45 1C A9
1C20  FD 8D 00 DF 2C 00 DF 70 06 20 5B 1D 4C B6 1A 20
1C30  38 1E 4C B6 1A 2C AC 17 30 0B AD B4 17 CD 96 17
1C40  D0 BE CE AC 17 20 5B 1D 4C B1 18 AE 96 17 AD 27
1C50  1E 69 01 A8 AD 28 1E 69 00 9D 00 01 CA 98 9D 00
1C60  01 CA 60 E8 BD 00 01 8D 27 1E E8 BD 00 01 8D 28
1C70  1E 8E 96 17 60 A2 18 20 F2 1D A9 00 8D 97 17 A2
1C80  08 20 EB FF C9 30 D0 05 0E 97 17 90 08 C9 31 D0
1C90  F0 38 2E 97 17 20 08 1E CA D0 E6 60 A2 2A 20 F2
1CA0  1D A0 00 B9 93 17 20 A7 1D 20 33 1E C8 C0 04 D0
1CB0  F2 A0 08 AD 97 17 8D AB 17 0E AB 17 B0 04 A9 30
1CC0  90 02 A9 31 20 08 1E 88 D0 EF 60 A2 32 20 F2 1D
```

```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
1CD0  AD 96 17 18 69 0B AA A0 0B BD 00 01 20 A7 1D 8C
1CE0  A8 17 20 C1 1D AC A8 17 AD 0F 1E 18 69 1E 8D 0F
1CF0  1E 90 03 EE 10 1E CA 88 D0 DF 60 A5 D0 EE D2 AD
1D00  FB 1C 8D 43 1D AD FC 1C 8D 44 1D A2 0A A0 12 4C
1D10  22 1D AD FD 1C 8D 43 1D AD FE 1C 8D 44 1D A2 07
1D20  A0 06 8C A7 17 AD 43 1D 8D 46 1D AD 44 1D 8D 47
1D30  1D AD 43 1D 18 69 20 8D 43 1D 90 03 EE 44 1D AC
1D40  A7 17 B9 FF FF 99 FF FF 88 10 F7 CA D0 D7 60 A9
1D50  FF 8D AC 17 8D AD 17 8D AE 17 60 20 FF 1C 20 3D
1D60  1E 20 9C 1C 4C CB 1C 8E A6 17 20 6D 1D 20 84 1D
1D70  0A 0A 0A 0A A2 04 2A 2E AB 17 CA D0 F9 AD AB 17
1D80  AE A6 17 60 20 EB FF C9 30 30 F9 C9 3A 90 08 C9
1D90  47 B0 F1 C9 41 90 ED 20 08 1E C9 41 90 02 E9 07
1DA0  29 0F 60 20 A7 1D 8A 8D AB 17 4A 4A 4A 4A 20 B4
1DB0  1D AD AB 17 29 0F 09 30 C9 3A 90 02 69 06 4C 08
1DC0  1E 48 AD AA 17 AC 0F 1E 8C D4 1D AC 10 1E 8C D5
1DD0  1D A0 20 99 FF FF 68 60 48 AD 0F 1E AC 10 1E 8D
1DE0  E8 1D 8C E9 1D A0 20 B9 FF FF 8D AA 17 A9 5E 4C
1DF0  C5 1D 8C A8 17 20 C1 1D BD B5 17 8D 0F 1E E8 BD
1E00  B5 17 8D 10 1E 4C 19 1E 8C A8 17 20 C1 1D 8D FF
1E10  FF EE 0F 1E D0 03 EE 10 1E 20 D8 1D AC A8 17 48
1E20  68 60 8D EC D2 60 B9 FF FF 60 AD 28 1E AE 27 1E
1E30  20 A3 1D A9 20 4C 08 1E EE AE 17 F0 08 A2 16 20
1E40  F2 1D 20 2A 1E A0 00 20 26 1E A8 4A 90 0B 4A B0
1E50  17 C9 22 F0 13 29 07 09 80 4A AA BD 21 1F B0 04
1E60  4A 4A 4A 4A 29 0F D0 04 A0 80 A9 00 AA 8E A9 17
1E70  BD 65 1F 8D B0 17 29 03 8D B1 17 2C AE 17 30 04
1E80  CE AE 17 60 98 29 8F AA 98 A0 03 E0 8A F0 0B 4A
1E90  90 08 4A 4A 09 20 88 D0 FA C8 88 D0 F2 48 20 26
1EA0  1E 20 A7 1D 20 33 1E 68 A8 B9 7F 1F 8D B2 17 B9
1EB0  BF 1F 8D B3 17 A2 03 A9 00 A0 05 0E B3 17 2E B2
1EC0  17 2A 88 D0 F6 69 3F 20 08 1E CA D0 EA 20 33 1E
1ED0  A2 06 E0 03 D0 1A AC B1 17 F0 15 A9 24 20 08 1E
1EE0  AD B0 17 C9 E8 20 26 1E B0 1D 20 A7 1D 88 D0 F0
1EF0  0E B0 17 90 0E BD 72 1F 20 08 1E BD 78 1F F0 03
1F00  20 08 1E CA D0 CC 60 20 13 1F AA E8 D0 01 C8 98
1F10  4C A3 1D AC 28 1E AA 10 01 88 6D 27 1E 90 01 C8
1F20  60 40 02 45 03 D0 08 40 09 30 22 45 33 D0 08 40
1F30  09 40 02 45 33 D0 08 40 09 40 02 45 B3 D0 08 40
1F40  09 00 22 44 33 D0 8C 44 00 11 22 44 33 D0 8C 44
1F50  9A 10 22 44 33 D0 08 40 09 10 22 44 33 D0 08 40
1F60  09 62 13 78 A9 00 21 01 02 00 80 59 4D 11 12 06
1F70  4A 05 1D 2C 29 2C 23 28 41 59 00 58 00 00 00 1C
1F80  8A 1C 23 5D 8B 1B A1 9D 8A 1D 23 9D 8B 1D A1 00
1F90  29 19 AE 69 A8 19 23 24 53 1B 23 24 53 19 A1 00
1FA0  1A 5B 5B A5 69 24 24 AE AE A8 AD 29 00 7C 00 15
1FB0  9C 6D 9C A5 69 29 53 84 13 34 11 A5 69 23 A0 D8
1FC0  62 5A 48 26 62 94 88 54 44 C8 54 68 44 E8 94 00
1FD0  B4 08 84 74 B4 28 6E 74 F4 CC 4A 72 F2 A4 8A 00
1FE0  AA A2 A2 74 74 74 72 44 68 B2 32 B2 00 22 00 1A
1FF0  1A 26 26 72 72 88 C8 C4 CA 26 48 44 44 A2 C8 24
```



The checksum at the end of each line is shown in decimal! If you set the MEMORY SIZE to 6032 you can use the following BASIC program to compare the checksums:

```
100 FOR I = 6032 TO 8191 STEP 16: SE = 0
   :FOR J = 0 TO 15
110 SE = SE + PEEK (I+J): NEXT J
120 PRINT SU; ", ",: NEXT I
```

Let's have a closer look at figure 1 and 2:
In the beginning the register had the following contents:
A=00, X=20, Y=4F, SP=FA, STATUS = 00100000.
The breakpoint B1 is set to the input routine.
The breakpoint B2 is set to the output routine.
The breakpoints 3 to 6 are deleted.
The addresses 7550-7555 with their contents were given out using the M-command. The return address-1 of subroutine-call in 1/2 is in the stack at 2/8 and 2/9. The saved contents of the accumulater is at 2/10, the contents of the X-register is at 2/11 and the contents of the Y-register is saved at 2/12.
In 1/10 the program jumps to ($7550), this means to $7300 as can be seen in block 6.
In 1/9 the accumulater has been loaded with 00, that for the zero-flag in 3/15 is set to "1" (the commands in 1/10 and 1/11 do not influence the zero-flag). After the execution of the command in 1/11 the programcounter will be at $7303, because the zero-flag is set to "1".
After five bytes having been put to the stack, the stackpointer now is at $F5.
At the end we want to point out that this debugger is a great help for the experienced assembler-programmer, as well as for the beginner, because the complete status of the CPU is shown after each command.

# Dogfight

DOGFIGHT

This game was written for a 64x32 screen but should work also on the 25x25 version (in which case the value for SV in line 22 may need to be increased by 32 or 64). The delay loop in line 2000 defines the speed of the game and may be changed.

```
 6 REM DOGFIGHT
 9 GOSUB8000
 10 FORT=1TO32:PRINT:NEXT
 22 SV=53376:OS=32:REM FOR C1P
 23 REM SV=53248:OS=64:REM FOR C4P
 25 POKE530,1:P7=SV+OS*18+5:P8=P7+OS-11:D1=1:
D2=4:P1=P7:P2=P8
 27 A(1)=237:A(4)=239:A(2)=236:A(3)=A(2):A(5)
=238:A(6)=A(5)
 30 KE=57088:F=0:F1=0
 32 FORX=SV+3TOSV+10*OS+3STEPOS:POKEX,43:POKE
X+OS-6,43:NEXT
 40 SR=0:SL=0:CO=0:FORX=SV+10*OS+3TOSV+23*OS+
3STEPOS
 50 POKEX,161:POKEX+OS-6,161:NEXT
 80 FORX=SV+23*OS+3TOSV+24*OS-4
 100 POKEX,161:NEXT:X=SV+17*OS+OS/2:XZ=0
 115 XZ=XZ+1:IFXZ>6THEN130
 120 POKEX,161:X=X+OS:GOTO115
 130 POKEKE,127:OP=PEEK(KE):POKEKE,251:OQ=PEE
K(KE)
```



```
 131 POKE530,0:POKE530,1
 133 SP=SL:P9=P7-17*OS:GOSUB400:SP=SR:P9=P8-1
7*OS-2:GOSUB400
 134 SP=CO:P9=P9+7-OS/2:GOSUB400
 136 CO=CO+1:IFCO<1000THEN 140
 137 INPUT"ANOTHER GO";A$
 138 IFLEFT$(A$,1)="Y"THEN10
 139 END
 140 IF(OP>233)AND(OQ>251)THEN160
 150 IFOP=127THEND1=D1+1
 151 IFOP=191THEND1=D1-1
 152 IF(OP=223)AND(F=0)THEN231
 153 IFD1>6THEND1=1
 154 IFD1<1THEND1=6
 155 IFOQ=239THEND2=D2+1
 156 IFOQ=247THEND2=D2-1
 157 IF(OQ=251)AND(F1=0)THEN301
 158 IFD2>6THEND2=1
 159 IFD2<1THEND2=6
 160 IFF=1THEN235
 162 IFF1=1THEN320
 164 PO=P1:PI=P2
 175 REM LEFT
 178 IF(P1+1=P2)OR(P1-1=P2)OR(P1+OS=P2)THENGO
SUB7000
 179 IF((PEEK(P1-1))OR(PEEK(P1+1)))<>32THEN40
00
 180 IFPEEK(P1+OS)<>32THEN4000
 185 GD=D1:GP=P1:GOSUB1000:P1=GP:IFF=0THENGOS
UB2000
 210 POKEPO,32:POKEP1,A(D1):GOTO281
 231 BD=D1:BP=P1:F=1:GD=BD:GP=BP:GOSUB1000:BP
=GP
 235 POKEBP,32
 236 W=0
 240 GD=BD:GP=BP:GOSUB1000
 245 LO=PEEK(GP):GOSUB6000:IFW>0THEN269
 250 GOSUB1000
 252 LO=PEEK(GP):GOSUB6000:IFW>0THEN269
 255 BP=GP:POKEBP,45:GOTO162
 269 F=0
 270 IFW=1THEN162
 271 GOTO5000
```

```
 280 REM RIGHT
 281 IF(P2-1=P1)OR(P2+1=P1)OR(P2+OS=P1)THENGO
SUB7000
 282 IF((PEEK(P2-1))OR(PEEK(P2+1)))<>32THEN50
00
 283 IFPEEK(P2+OS)<>32THEN5000
 285 GD=D2:GP=P2:GOSUB1000
 287 P2=GP:IFF1=0THENGOSUB2000
 295 POKEPI,32:POKEP2,A(D2)
 297 GOTO130
 301 BE=D2:BX=P2:F1=1:GD=BE:GP=BX:GOSUB1000:B
X=GP
 320 POKEBX,32
 325 W=0
 340 GD=BE:GP=BX:GOSUB1000
 345 LO=PEEK(GP):GOSUB6000:IFW>0THEN360
 350 GOSUB1000
 352 LO=PEEK(GP):GOSUB6000:IFW>0THEN360
 355 BX=GP:POKEBX,46:GOTO164
 360 F1=0
 365 IFW=1THEN164
 370 GOTO4000
 400 A$=STR$(SP):POKEP9,ASC(RIGHT$(A$,3))
 410 POKEP9+1,ASC(RIGHT$(A$,2))
 420 POKEP9+2,ASC(RIGHT$(A$,1)):RETURN
 999 REM DIRECTION
 1000 IFGD=1THENGP=GP+1:RETURN
 1010 IFGD=2THENGP=GP-OS+1:RETURN
 1020 IFGD=3THENGP=GP-OS-1:RETURN
 1030 IFGD=4THENGP=GP-1:RETURN
 1040 IFGD=5THENGP=GP+OS-1:RETURN
 1050 IFGD=6THENGP=GP+OS+1:RETURN
 2000 FORDE=0TO10:NEXTDE:RETURN
 3999 REM CRASH
 4000 SR=SR+1
 4005 FORZ=1TO3:POKEP1,Z:NEXTZ:POKEP1,32
 4040 IFPEEK(P1+OS)<>32THEND1=1:P1=P7:GOTO130
 4045 P1=P1+OS:GOTO4005
 5000 SL=SL+1
 5005 FORZ=1TO3:POKEP2,Z:NEXTZ:POKEP2,32
 5040 IFPEEK(P2+OS)<>32THEND2=4:P2=P8:GOTO130
 5045 P2=P2+OS:GOTO5005
 6000 IF((LO>235)AND(LO<240))THENW=2:RETURN
```



```
 6010 IFLO<>32THENW=1:RETURN
 6020 RETURN
 7000 FORZ=1TO3:POKEP2,Z:POKEP1,Z:NEXTZ:POKEP
1,32:POKEP2,32
 7040 IFPEEK(P2+OS)<>32THEND2=4:D1=1:P2=P8:P1
=P7:RETURN
 7050 P2=P2+OS:P1=P1+OS:GOTO7000
 8000 PRINT"  DOGFIGHT !  "
 8005 PRINT
 8010 PRINT"THE OBJECT  OF THIS GAME
 8015 PRINT"IS  TO  SHOOT  DOWN  THE
 8020 PRINT"OTHER PLAYERS PLANE WITH
 8025 PRINT"A MISSILE.
 8030 PRINT"A  POINT IS  SCORED  FOR
 8035 PRINT"EACH SUCCESFUL HIT.
 8040 PRINT"THE OPPONENT ALSO GETS A
 8045 PRINT"POINT IF YOU CRASH.
 8050 PRINT"MIDAIR COLLISIONS RESULT
 8055 PRINT"IN  ZERO SCORE  FOR BOTH
 8060 PRINT"PLAYERS.
 8070 PRINT"THE CONTROLS ARE:
 8080 PRINT"LEFT <--        --> RIGHT
 8090 PRINT"(1) PLANE TURNS LEFT (B)
 8100 PRINT"(2) PLANE TURNS RIGHT(N)
 8110 PRINT"(3) FIRES A MISSILE  (M)
 8120 PRINT"LEFT --> (1) TO TAKE OFF
 8130 PRINT"RIGHT--> (N) TO TAKE OFF
 8140 PRINT:INPUT"READY (Y/N)";A$
 8150 IFLEFT$(A$,1)="Y"THEN RETURN
 8160 GOTO8000
OK
```

# Start & landing simulation

START & LANDING SIMULATION

1. Description of problem and answer

A given quantity is the mass m (space ship) in the gravitational field. The program approximately calculates the orbit of m depending on the start conditions.
The player can change the following parameters during the running program:
1.) Gradient. Angle of flight direction to surface of earth.
2.) Engine on/off. Change of speed and direction.
3.) Thrust. Power of engine.

The above problem can be described by the following differential equation:

$$m \cdot \frac{d^2}{dt^2} \cdot Vx(t) - \frac{\delta}{\delta rx} \cdot \frac{f \cdot M \cdot m}{r(t)} = 0$$

$$m \cdot \frac{d^2}{dt^2} \cdot ry(t) - \frac{\delta}{\delta ry} \cdot \frac{f \cdot M \cdot m}{r(t)} = 0$$

$$\text{with } r(t) = \sqrt{rx^2(t)\ try^2(t)}$$

rx,ry are the cartesian coordinates of m.
f is the gravitation constant by Newton.
M is the gravitating mass at the origin of the co-ordinate
wanted are rx(t) and ry(t).
rz(t), the third co-ordinate can be set to 0, because a mass that is affected only by a central force always moves in a plane (in this case the x,y-plane). Of course,



accellerations in z-direction (by rocket-engine) then have to be avoided.

We use the approximation that the potential

$$\frac{f \cdot M \cdot m}{r(t)}$$

is constant for short times $\Delta t$.

Double integration of (I) gives:

$$rx = -\frac{f \cdot M \cdot rx}{\delta \cdot r^3} \cdot \Delta t^2 + Vx \cdot \Delta t + ax$$

$$ry = -\frac{f \cdot M \cdot rx}{\delta \cdot r^3} \cdot \Delta t^2 + Vy \cdot \Delta t + ay$$

Where Vx,Vy,ax,ay are integration constants.
With $\Delta t$ = 1 sec and $\alpha$ = f.M we get orbit points in distances of $\Delta t$ = 1 sec:

$$rx(t+\Delta t) = -\frac{\alpha}{2} \cdot \frac{rx(t)}{r^3} + Vx(t) + rx(t) \quad \text{(II a)}$$

$$ry(t+\Delta t) = -\frac{\alpha}{2} \cdot \frac{ry(t)}{r^3} + Vy(t) + ry(t)$$

For the speeds vx,vy we get by one time integration of (I):

$$Vx(t+\Delta t) = -\alpha \cdot \frac{rx(t)}{r^3} + Vx(t)$$

$$Vy(t+\Delta t) = -\alpha \cdot \frac{ry(t)}{r^3} + Vy(t) \quad \text{(II b)}$$

Changes of speed by thrust are discribed by:

$$Vx = Vx + Sx$$
$$Vy = Vy + Sy$$

Where sx,sy are functious of thrust and gradient.

2. Screen-output

rx,ry,vx,vy are not directly decisive for control of the flight and are therefore not shown. Continously shown, however, are horizontal and vertical speed related to the surface of earth (vh and vv), as well

as the height H.

$$VH(t) = (rx(t) \cdot Vx(t) + ry(t) \cdot vy(t)) \cdot \frac{1}{r(t)} \qquad \text{where } r(t) = \sqrt{rx^2(t) + ry^2(t)}$$
$$Vv(t) = (rx(t) \cdot Vy(t) - ry(t) \cdot vx(t)) \cdot \frac{1}{r(t)} \qquad \text{and } R = \text{radius of the earth} \qquad \text{(III)}$$
$$H(t) = r(t) - R$$

In addition to that gradient, thrust, contents of tank and the time are displayed.

3. Run of the program

After the program description the program asks you "start-or landing-simulation (S/L)?"
If you enter "L", the program simulates the landing.
The space ship moves in a height of 100km with a horizontal speed of 8097 m/sec (orbit). Over and over again the program calculates the new values for rx,ry,vx,vy using equations (11) and vh,vv,H using equations (111), and constantly displays these values.

The keys 1-6 are for control:
1,2 change the gradient; 3,4 change the thrust; 5,6 turn the engine on or off (blinking square = on).
Further start conditions are:
vv(0)=0; thrust=0; engine=off; gradient=0; content of tank=1100 l.

If you enter "S" the start conditions are:
vv=0; vh=0; thrust=50; gradient=90° ; ry(0)=radius of earth; rx(0)=0; content of tank = 2500 l; engine = on!
This means it is on accellerated flight up. You should reach a constant orbit with vv=0; The speed vh (dependent of height) necessary for that will be displayed if you hit key 7. $(VH = \sqrt{\frac{f \cdot m}{r}})$



The landing graphics (subroutine 1100 - 1140) show the last 1000 m above the surface of earth (right side of screen).
4. Program description

| | |
|---|---|
| 10 - 18 | explanation |
| 60 - 100 | display |
| 130 - 170 | start values and constants |
| 200 - 370 | orbit and speed calculations |
| 380 - 411 | reading of keyboard |

Subroutines

| | |
|---|---|
| 450 | end of program, restart |
| 500 - 530 | engine on; change of speed, fuel |
| 600 - 850 | change of gradient |
| 900 - 1010 | change of thrust |
| 1100 - 1140 | landing graphics |
| 2000 - 2030 | calculation of height |
| 5000 | output of strings to screen |

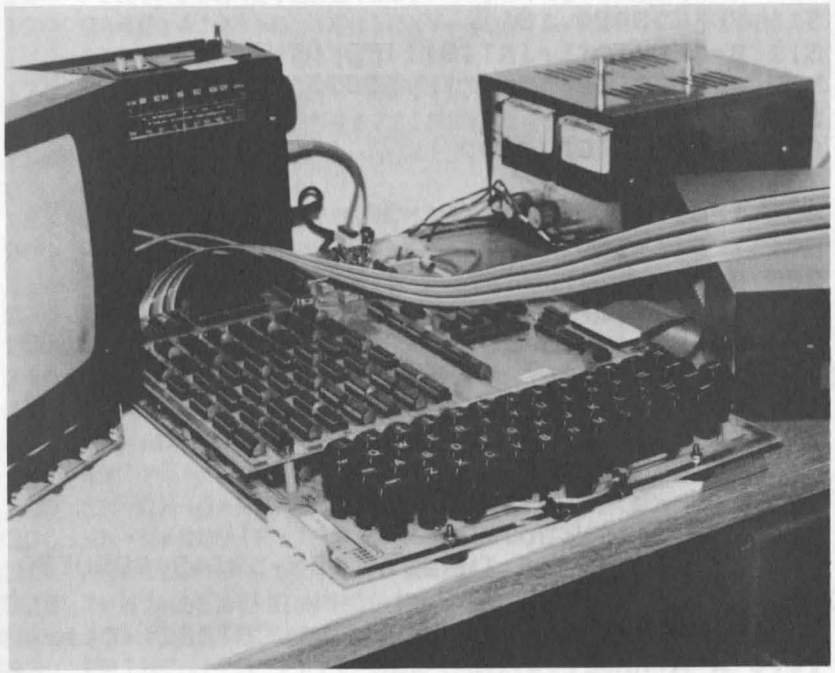
```
 1 REM   SIMULATION-PROGRAM
 2 REM   ELCOMP PUBLISHING INC.
 3 REM
 4 REM
 5 REM*****************************************
****
 6 REM
 10 PRINT"************************"
 11 PRINT"START&LANDINGSIMULATION "
 12 PRINT"************************"
 13 PRINT:PRINT:PRINT"  CONTROL":PRINT
 14 PRINT" 1,2 = -,+ ANGLE
 15 PRINT" 3,4 = -,+ THRUST
 16 PRINT" 5,6 = ON,OFF ENGINE
 17 PRINT" 7   = V-HOR.FOR CIRCLE
 18 FORI=1TO5:PRINT:NEXT
 55 INPUT" START-OR LANDING-      SIMULATI
ON(S/L)";GG$
 56 FORI=1TO30:PRINT:NEXT
 57 GG=1:IFGG$="L"THENGG=2
```

```
 58 POKE530,1:POKE57088,127
 60 A$="GRADIENT":X=53639:GOSUB5000
 65 A$="HORIZONTAL":X=53831:GOSUB5000
 70 A$="VERTICAL":X=53895:GOSUB5000
 80 A$="ENGINE         TANK":X=53478:GOSUB50
00
 90 A$="HEIGHT        TIME":X=54023:GOSUB500
0
 95 FORI=53498TO54106STEP32:POKEI,146:NEXT
 96 FORI=54138TO54140:POKEI,145:NEXT
 97 FORI=53571TO53593:POKEI,161:NEXT
 98 FORI=53699TO53721:POKEI,161:NEXT
 99 FORI=53955TO53977:POKEI,161:NEXT
 100 FORI=53342TO53436:POKEI,187:NEXT
 130 G=4E14:RY=6.1E6:CX=-SQR(G/RY):B=1100
 160 ONGG GOTO170,195
 170 B=2510:K1=1:RY=6E6:CX=0:Q1=90:W=50
 190 FORI=54147TO54172:POKEI,161:NEXTI
 191 A$="ORBIT AT V-HORIZ.":X=54182:GOSUB50
00
 195 GOSUB500:GOSUB807:GOSUB900
 200 R=SQR(RX^2+RY^2)
 210 IFRX>0THENA=ATN(RY/RX):GOTO250
 220 IFRX<0THENA=3.14+ATN(RY/RX):GOTO250
 230 IFRY>0THENA=1.57:GOTO250
 240 A=-1.57
 250 F=A+Q-1.57
 260 RX=-G*RX/(2*R^3)+CX+RX:RY=-G*RY/(2*R^3
)+CY+RY
 280 CX=CX-G*RX/R^3:CY=CY-G*RY/R^3
 300 SX=W*COS(F):SY=W*SIN(F):T1=T1+1
 320 POKE53490,32:H=R-6E6
 326 IFH>=1E6THENPRINTH/1000;"KM",T1;"SEC";
CHR$(13);:GOTO332
 330 PRINTH;"M",T1;"SEC";CHR$(13);
 332 IFH<1000THENGOSUB1100
 337 IFH<0ANDGG=0THENPRINT:PRINT" L A N D E
 D":GOTO450
 340 CH=(RX*CX+RY*CY)/R:CV=(RX*CY-RY*CX)/R
 360 A$=STR$(CH)+" M/SEC    ":X=53927:GOSUB5
000
 370 A$=STR$(CV)+" M/SEC    ":X=53863:GOSUB5
000
```



```
 380 K=PEEK(57088)
 390 IFK1=1THENGOSUB500
 400 IFK=127THENGOSUB600
 405 IFK=191THENGOSUB700
 406 IFK=223THENGOSUB900
 407 IFK=239THENGOSUB1000
 408 IFK=247THENK1=1
 409 IFK=251THENK1=0
 410 IFK=253THENGOSUB2000
 411 GOTO200
 450 FORT=0TO5000:NEXT:FORT=53300TO54300:PO
KET,32:NEXT:RUN55
 500 IFB=0THENRETURN
 510 CX=CX+SX:CY=CY+SY:POKE53490,161
 511 POKE53490,161
 512 B=B-W/10:B=INT(B):IFB<0THENB=0
 520 A$=STR$(B)+"L ":X=53555:GOSUB5000
 530 RETURN
 600 Q1=Q1-5:GOTO800
 700 Q1=Q1+5
 800 IFQ1>180THENQ1=Q1-360
 801 IFQ1<-180THENQ1=Q1+360
 805 GG=0
 807 Q=3.1416/180*Q1
 810 A$=STR$(Q1)+" DEGREES ":X=53671:GOSUB5
000
 830 Q2=(Q1+180)/45:Q2=INT(Q2)
 840 IFQ2>5THENQ2=Q2-8
 850 POKE53655,250+Q2:RETURN
 900 W=W-XX:IFW<0THENW=0
 901 XX=10:IFW<30THENXX=5
 910 A$=STR$(W)+" THRUST   ":X=53543:GOSUB50
00
 940 RETURN
 1000 W=W+XX
 1001 XX=10:IFW<30THENXX=5
 1010 GOTO910
 1100 Z=INT(H/50)+1:Z1=INT((H-50*INT(H/50))
*7/49+.5)
 1120 FORI=53499TO54107STEP32:POKEI,32:NEXT
 1130 POKE54139-32*Z,128+Z1
 1140 RETURN
 2000 V2=SQR(G/R)
```

```
 2010 A$=STR$(V2)+" M/SEC      ":X=54214:GOS
UB5000
 2030 RETURN
 5000 FORI=1TO LEN(A$):POKEI+X,ASC(MID$(A$,
I,1)):NEXTI:RETURN
OK
```





# Drawing on the screen with C4P

DRAWING ON THE SCREEN WITH C4P

The following program allows you to draw on the screen with your C4P.

After you have started the program it will ask for the background color. Enter a number between 0 and 15. Even numbers from 0 to 14 cause a dark background , odd numbers from 1 to 15 cause a light background. Then select a character by typing in three numbers (000...255). If you enter 032 or 999 the character will be a blank.
The character will be displayed in the lower lefthand corner. If you enter CTRL and two numbers (00...15) you define the color of the character. The color and character can be changed while drawing. The keys around -H- define the direction for the CURSOR. The direction is shown by an arrow in the lower lefthand corner of the screen and can also be changed.
If you now type H the CURSOR will appear and, if you hit H again, it will move in the selected direction.
If you want to save the picture on tape you have to enter SHIFT-S. You will see a S in the lower lefthand corner. If the S is replaced by an E you can turn off the recorder.

```
 10 FOR I=1 TO 32:PRINT:NEXT
 20 PRINT"   DRAWING ON THE COLOR TV
 25 PRINT
 26 PRINT"   1.       SELECT CHARACTER (000..
.255)
 27 PRINT
 28 PRINT"   2.       SELECT COLOR(CTRL00...C
TRL15)
 29 PRINT
 30 PRINT"   3.       SELECT DIRECTION
 31 PRINT"            WITH KEYS AROUND -H-
 32 PRINT
 33 PRINT"   4.    MOVE CURSOR WITH -H- OR
 35 PRINT"          DRAW CHARACTER AND COLOR
 36 PRINT"          WITH -SHIFT H-
 37 PRINT
 46 PRINT"   5. SAVE PICTURE WITH -SHIFT S-
 47 PRINT
 48 PRINT"   6. LOAD PICTURE WITH -SHIFT L-
 49 PRINT:PRINT:PRINT
```



```
 50 SA=53248:SE=55295:CO=4096:Q=64512:R=Q+1
 60 INPUT"BACKGROUND COLOR";FA
 65 IF FA>15 THEN FA=15
 70 FOR I=SA TO SE:POKE I,32:POKE I+CO,FA:NE
XT
 80 SA=53312:SC=54160:J=55172:H=55168:Z=0:T=
0:
 90 POKE 15,64:POKE56832,7
 100 POKE 11,0:POKE 12,253
 110 X=USR(X)
 120 K=PEEK(57100):P=PEEK(531)
 130 IF P=88 THEN 400 REM SHIFT H - DRAW
 140 IF P=72 THEN 450 REM MOVE CURSOR
 150 IF P<58 THEN 300 REM 000...255 - CHARAC
TER
 160 IFP>236 THEN 500 REM CTRL00...15 = COLO
R
 170 IF P=99 THEN GOSUB 700  REM SAVE WITH -
SHIFT S-
 175 IF P=90 THEN 900
 180 IF P=92 THEN GOSUB 750  REM LOAD WITH -
SHIFT L-
 185 IF P=86 THEN 950
 190 REM CURSOR
 200 IF P=89 THEN V=-64:CU=16:GOTO 600
 205 IF P=85 THEN V=-63:CU=17:GOTO 600
 210 IF P=74 THEN V=  1:CU=18:GOTO 600
 215 IF P=77 THEN V= 65:CU=19:GOTO 600
 220 IF P=78 THEN V= 64:CU=20:GOTO 600
 225 IF P=66 THEN V= 63:CU=21:GOTO 600
 230 IF P=71 THEN V= -1:CU=22:GOTO 600
 235 IF P=84 THEN V=-65:CU=23:GOTO 600
 240 GOTO 110
 290 REM SELECT CHARACTER
 300 IF P<48 THEN 110
 305 P=P-48:Z=Z+1:ON Z GOTO 310,320,330
 306 Z=0:GOTO 110
 310 E=P:E=E*100:GOTO 110
 320 F=P:F=F*10:GOTO 110
 330 G=P:T=E+F+G:IF T>255 THEN T=32
 350 POKE H+1,T:Z=0:GOTO 110
 390 REM CHARACTER AND COLOR TO SCREEN
 400 POKE SC,T:POKE SC+CO,FA:GOTO 460
```

```
 440 REM MOVE CURSOR ONLY
 450 POKE SC+CO,B
 460 SC=SC+V:IF SC>SE THEN SC=SC-V
 470 IF SC<SA THEN SC=SC-V
 480 B=PEEK(SC+CO):POKE SC+CO,B+1:GOTO 110
 490 REM COLOR = 2 NUMBERS (CTRL00...CTRL15)
 500 P=P-240:Y=Y+1:ON Y GOTO 510,520
 510 W=P:W=W*10:GOTO 110
 520 U=P:FA=U+W: REM DISPLAY COLOR
 530 FOR I=H TO J:POKE I+CO,FA:NEXT
 540 Y=0:GOTO 110
 590 REM DISPLAY CURSOR DIRECTION
 600 POKE J-1,CU:GOTO 110
 690 REM SAVE PICTURE ON TAPE
 700 POKE J+1,83:FOR I=1TO255:WAITQ,2:POKER,
1:NEXT
 710 FOR L=SA TO SE:T=PEEK(L):WAIT Q,2:POKE
R,T:NEXT
 720 FOR L=SA+CO TO SE+CO:T=PEEK(L):WAIT Q,2
:POKE R,T:NEXT
 730 POKE J+1,69:RETURN
 740 REM LOAD PICTURE FROM TAPE
 750 SC=SA
 755 WAITQ,1:T=PEEK(R):IFT=1 THENSC=SC+1
 760 POKESC,T:IFSC>SA+50THENSC=SA:GOTO800
 770 GOTO755
 800 WAITQ,1:T=PEEK(R):IFT=1THENSC=SA:GOTO80
0
 810 POKESC,T:SC=SC+1:IFSC>SETHENSC=SA+CO-1:
GOTO830
 820 WAITQ,1:T=PEEK(R):GOTO810
 830 POKESC,T:SC=SC+1:IFSC>SE+COTHEN850
 840 WAITQ,1:T=PEEK(R):GOTO830
 850 SC=54160:RETURN
 900 POKE 57089,21:GOTO 110
 950 POKE 57089,6:GOTO110
OK
```



# Tank war game for dual-joystick

TANK WAR GAME FOR DUAL-JOYSTICK

Load the program with LOAD/RUN, type in a random number to control the right tank from the keyboard, press SHIFT/LOCK.

Tank (right side)

Key 9 equals to move the tank
Key 0 turns the tank around
Key 8 fire

Tank (left side)

Key 2 moves the tank
Key 3 turns the tank around (direction)
Key 1 fire

TANKWAR FOR SUPERBOARD 11

```
 4 REM * ELCOMP PUB USA*
 5 REM ******************
 6 FORI=1TO35:PRINT:NEXT:GOSUB12001:INPUT"RA
NDOM #";SE
 7 RNGE=10:PL=2:DEFFNM8(X)=X+8-8*INT((X+7)/8
)
 8 FORI=1TO8:READVEC(I):NEXT I
 9 GOSUB12000:GOSUB13000
 10 PN(1)=53849:PN(2)=53831
```

```
11 DIR(1)=7:DIR(2)=3
12 KEY=0:GOSUB10000:PL=1:GOSUB10000
13 FORI=1TO1500:NEXTI
14 KEY=5:GOSUB10000:PL=2:GOSUB10000
20 GOSUB10000:GOSUB20000
22 IF(FIRE(1)=-1 OR FIRE(2)=-1)THEN GOTO30
23 IF PL=1 THEN PL=2:GOTO 20
24 PL=1
25 GOTO 20
30 FORI=1TO1500:NEXTI
31 GOSUB12001
32 INPUT"AGAIN";AG$:IFLEFT$(AG$,1)="Y"THEN9
33 FOR I=1TO35:PRINT"":NEXT
34 PRINT"   COWARDS, BOTH OF YOU"
35 FORI=1TO35:PRINT"":NEXT:END
10000 REM**EXECUTION OF PLAYER INSTRUCTIONS
**
10020 IF FIRE(PL)>0THEN10400
10040 POKE PN(PL),32
10060 LOC=PN(PL)
10080 IF KEY=1 THEN DIR(PL)=FNM8(DIR(PL)+1)
10100 IF KEY=2 THEN DIR(PL)=FNM8(DIR(PL)-1)
10120 IF KEY=3 THEN GOSUB 21000
10140 IF KEY<>4 THEN GOTO10240
10160 CC=DIR(PL)
10180 DIR(PL)=FNM8(CC-4)
10200 GOSUB21000
10220 DIR(PL)=CC
10240 POKE LOC,247+DIR(PL)
10250 PN(PL)=LOC
10260 IF KEY<>5 THEN 10380
10270 IF(FIRE(PL)>0)THEN10380
10280 BP(PL)=PN(PL)
10290 DB(PL)=DIR(PL)
10300 FIRE(PL)=1
10340 BP(PL)=LOC
10350 CD=DIR(PL)
10360 GOTO10420
10380 GOTO10700
10400 POKE BP(PL),32
10420 LOC=BP(PL)
10440 CE=LOC
10450 CD=DIR(PL):DIR(PL)=DB(PL)
```



```
10460 GOSUB21000
10470 IF CE=LOC THENFIRE(PL)=0:DIR(PL)=CD:R
ETURN
10480 RA=PEEK(LOC)
10490 BP(PL)=LOC
10500 IF(247<RA AND RA<256)THEN 10600
10520 FIRE(PL)=FIRE(PL)+1
10525 IF(RA<>13 AND RA<>188)THEN 10540
10530 FORI=0TO 255:POKE LOC,I:NEXT
10535 FIRE(PL)=0:POKE LOC,32
10536 DIR(PL)=CD
10537 RETURN
10540 POKE LOC,15+DIR(PL)
10550 DIR(PL)=CD
10560 IF FIRE(PL)<RNGETHEN10590
10580 FIRE(PL)=0
10585 POKE(LOC),(32)
10590 REM
10592 GOTO10630
10600 FIRE(PL)=-1
10620 FORI=1TO255:POKE LOC,I:NEXT
10625 POKE LOC,32
10627 RETURN
10630 GOTO10040
10700 RETURN
12000 GOSUB12001:GOTO12005
12001 FORI=54108TO54300:POKEI,32:NEXTI
12002 RETURN
12005 FORI=1TO35:PRINT"":NEXT
12010 PRINT"      TANK WAR MK I"
12020 FORI=1TO8:PRINT"":NEXT
12030 RETURN
13000 REM TREES AND HILLS
13050 CD=13
13075 FORI=1TO80*RND(SE)+1
13100 UA=  53378+INT(1023*RND(SE))
13200 GOSUB15000
13250 NEXT I
13300 FORI=1TO5
13400 UA=53378+INT(1023*RND(SE))
13500 CD=188:GOSUB15000
13600 FORJ=1TO1+80*RND(SE)
13700 UA=UA+VEC(1+INT(8*RND(SE)))
```

```
13800 GOSUB15000
13900 NEXT J:NEXT I
13925 RETURN
15000 DX=UA-53379-INT((UA-53379)/32)*32
15100 IF(DX>26)THENRETURN
15110 IFDX<3ORDX>25THENRETURN
15200 POKEUA,CD:RETURN
20000 REM***READS KEYBOARD
20100 IFPL=1THENRA=127:GOTO20200
20150 RA=191
20200 POKE530,1:POKE57088,RA
20250 KEY=0
20260 CA=PEEK(57088)
20270 IFCA=247THENKEY=1
20280 IFCA=223THENKEY=2
20290 IFCA=191THENKEY=3
20300 IFCA=239THENKEY=4
20310 IFCA=127THENKEY=5
20320 IFCA=251THENKEY=6
20330 IFCA=253THENKEY=7
20500 POKE 530,0
20600 RETURN
21000 REM COMPUTE NEW LOCATION AND CHECK FO
R L
21150 CA=LOC+VEC(DIR(PL))
21175 IFPEEK(CA)=188THENRETURN
21200 RA=CA-53379-INT((CA-53379)/32)*32
21300 RB=LOC-53379-INT((LOC-53379)/32)*32
21400 IF ABS(RA-RB)>5THEN RETURN
21450 IFRA<2ORRA>25THENRETURN
21500 IF(CA<53379ORCA>54268)THENRETURN
21600 LOC=CA
21700 RETURN
30000 DATA -32,-31,1,33,32,31,-1,-33,53379,
541
OK
```



## TANKWAR FOR SUPERBOARD 111 (WITH SOUND)

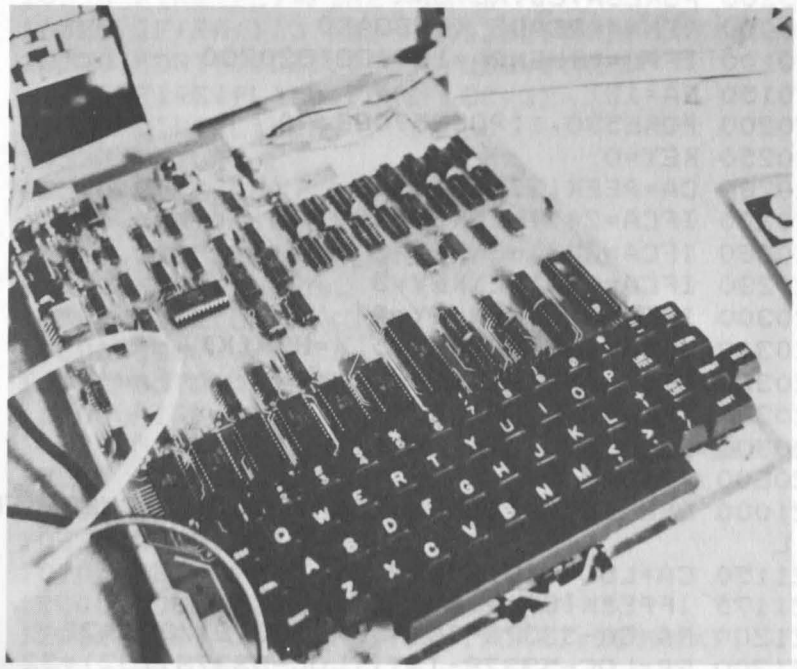
```
 3 FORX=0TO78:READW
 4 POKE560+X,W:NEXTX
 5 POKE11,48:POKE12,2
 6 FORI=1TO35:PRINT:NEXT:GOSUB12001:INPUT"RA
NDOM #";SE
 7 RNGE=10:PL=2:DEFFNM8(X)=X+8-8*INT((X+7)/8
)
 8 FORI=1TO8:READVEC(I):NEXT I
 9 GOSUB12000:GOSUB13000
 10 PN(1)=53849:PN(2)=53831
 11 DIR(1)=7:DIR(2)=3
 12 KEY=0:GOSUB10000:PL=1:GOSUB10000
 13 FORI=1TO1000:NEXT
 14 KEY=5:GOSUB10000:PL=2:GOSUB10000
 20 GOSUB10000:GOSUB20000
 22 IF(FIRE(1)=-1 OR FIRE(2)=-1)THEN GOTO30
 23 IF PL=1 THEN PL=2:GOTO 20
 24 PL=1
 25 GOTO 20
 30 FORI=1TO1500:NEXTI
 31 GOSUB12001
 32 INPUT"AGAIN";AG$:IFLEFT$(AG$,1)="Y"THEN9
 33 FOR I=1TO35:PRINT"":NEXT
 34 PRINT"   COWARDS, BOTH OF YOU"
 35 FORI=1TO35:PRINT"":NEXT:END
 10000 REM**EXECUTION OF PLAYER INSTRUCTIONS
**
 10020 IF FIRE(PL)>0THEN10400
 10040 POKE PN(PL),32
 10060 LOC=PN(PL)
 10080 IF KEY=1 THEN DIR(PL)=FNM8(DIR(PL)+1)
 10100 IF KEY=2 THEN DIR(PL)=FNM8(DIR(PL)-1)
 10120 IF KEY=3 THEN GOSUB 21000
 10140 IF KEY<>4 THEN GOTO10240
 10160 CC=DIR(PL)
 10180 DIR(PL)=FNM8(CC-4)
 10200 GOSUB21000
 10220 DIR(PL)=CC
 10240 POKE LOC,247+DIR(PL)
 10250 PN(PL)=LOC
 10260 IF KEY<>5 THEN 10380
```

```
10270 IF(FIRE(PL)>0)THEN10380
10280 BP(PL)=PN(PL)
10290 DB(PL)=DIR(PL)
10300 FIRE(PL)=1
10340 BP(PL)=LOC
10350 CD=DIR(PL)
10360 GOTO10420
10380 GOTO10700
10400 POKE BP(PL),32
10420 LOC=BP(PL)
10440 CE=LOC
10450 CD=DIR(PL):DIR(PL)=DB(PL)
10460 GOSUB21000
10470 IF CE=LOC THENFIRE(PL)=0:DIR(PL)=CD:R
ETURN
10480 RA=PEEK(LOC)
10490 BP(PL)=LOC
10500 IF(247<RA AND RA<256)THEN 10600
10520 FIRE(PL)=FIRE(PL)+1
10525 IF(RA<>13 AND RA<>188)THEN 10540
10530 FORI=0TO255:POKE LOC,I:NEXT
10535 FIRE(PL)=0:POKE LOC,32
10536 DIR(PL)=CD
10537 RETURN
10540 POKE LOC,15+DIR(PL)
10550 DIR(PL)=CD
10560 IF FIRE(PL)<RNGETHEN10590
10580 FIRE(PL)=0
10585 POKE(LOC),(32)
10590 REM
10592 GOTO10630
10600 FIRE(PL)=-1
10620 FORI=1TO255:POKE LOC,I:NEXT:X=USR(X)
10625 POKE LOC,32
10627 RETURN
10630 GOTO10040
10700 RETURN
12000 GOSUB12001:GOTO12005
12001 FORI=54108TO54300:POKEI,32:NEXTI
12002 RETURN
12005 FORI=1TO35:PRINT"":NEXT
12010 PRINT"     TANK WAR MK I"
12020 FORI=1TO8:PRINT"":NEXT
```



```
12030 RETURN
13000 REM TREES AND HILLS
13050 CD=13
13075 FORI=1TO80*RND(SE)+1
13100 UA=  53378+INT(1023*RND(SE))
13200 GOSUB15000
13250 NEXT I
13300 FORI=1TO5
13400 UA=53378+INT(1023*RND(SE))
13500 CD=188:GOSUB15000
13600 FORJ=1TO1+80*RND(SE)
13700 UA=UA+VEC(1+INT(8*RND(SE)))
13800 GOSUB15000
13900 NEXT J:NEXT I
13925 RETURN
15000 DX=UA-53379-INT((UA-53379)/32)*32
15100 IF(DX>26)THENRETURN
15110 IFDX<3ORDX>25THENRETURN
15200 POKEUA,CD:RETURN
20000 REM***READS KEYBOARD
20100 IFPL=1THENRA=127:GOTO20200
20150 RA=191
20200 POKE530,1:POKE57088,RA
20250 KEY=0
20260 CA=PEEK(57088)
20270 IFCA=247THENKEY=1
20280 IFCA=223THENKEY=2
20290 IFCA=191THENKEY=3
20300 IFCA=239THENKEY=4
20310 IFCA=127THENKEY=5:X=USR(X)
20320 IFCA=251THENKEY=6
20330 IFCA=253THENKEY=7
20500 POKE 530,0
20600 RETURN
21000 REM COMPUTE NEW LOCATION AND CHECK FO
R L
21150 CA=LOC+VEC(DIR(PL))
21175 IFPEEK(CA)=188THENRETURN
21200 RA=CA-53379-INT((CA-53379)/32)*32
21300 RB=LOC-53379-INT((LOC-53379)/32)*32
21400 IF ABS(RA-RB)>5THEN RETURN
21450 IFRA<2ORRA>25THENRETURN
21500 IF(CA<53379ORCA>54268)THENRETURN
```

```
 21600 LOC=CA
 21700 RETURN
 25000 DATA169,255,141,128,2,133,26,169,16,1
41
 25010 DATA0,216,32,102,2,165,26,141,0,223
 25020 DATA32,95,2,169,0,141,0,223,32,95
 25030 DATA2,206,128,2,208,6,169,0,141,0
 25040 DATA216,96,198,26,76,60,2,174,127,2
 25050 DATA202,208,253,96,216,56,165,19,101,
22
 25060 DATA101,23,133,18,162,4,181,18,149,19
 25070 DATA202,16,249,165,18,141,127,2,96
 30000 DATA -32,-31,1,33,32,31,-1,-33,53379,
541
OK
```





# Paging on the superboard

PAGING ON THE SUPERBOARD

Most of the less expensive microcomputers don't have the feature of shiftable screen-memory.
In this chapter we will show you how to create a second video memory in the normal RAM area. The disadvantage of this technique is that you lose 1K of your workspace; but very often you don't need the whole memory anyway.
The program is written for the OSI Superboard, where the video memory is at $D000-$D3FF (1024 bytes). The contents of this memory is displayed continously on the screen. We will define another area of memory to be the screen memory. Then we can switch between the two pages by exchanging the contents of the two areas. The page that is not displayed at the moment is stored in normal memory. The two pages are independent, which means if something slips out of one page it will not be added at the bottom of the other.
The principle of the program is simple: a subroutine in machine language simply exchanges byte-by-byte between the video memory from $D000 to $D3FF and the workspace from $1C00 to $1F00.

The machine language routine does this job so fast that it looks like an instant switching between the two screens.
The machine language routine was developed for the 6502 CPU. Here is how it works:
First the index register X is erased. It will be used as an additional address added to a base-address. The accumulator will be loaded with the contents of the address $D000+X, which is the first address of the screen memory. Then the Y-register will be loaded with the contents of the address $1C00+X and the contents of the accu will be placed in that address. Now the first byte has been carried from screen memory to work memory. Next the contents of the Y-register is moved to the accumulator (command TYA) and then the contents of the accumulator is stored at location $D000+X.
The exchange of the other 1023 bytes is very similar. The only problem is that the X-register is one byte wide and can be used only for 256 running addresses. That is why the whole thing is done four times with four different base addresses. At the end of the program is an RTS-command that brings us back to BASIC.
The machine-language routine is at memory locations 574 (dec.) to 631 (dec.) which are normally free. The routine is relocatable.
The routine is called from BASIC by X=USR(X). Before the call, the start address of the program has to be stored in address 11 (dec., low byte) and 12 (dec., high byte).The sample program uses a second machine-language routine to clear the screen.
In line 50 the top of memory is redefined because the upper 1k of RAM is needed for the second screen page. At the end of the program (line 410) the pointer to top of memory is set back to its original value.



In lines 100-150 the machine language routine for exchange of the two memory areas is loaded. In lines 160-170 the machine language routine for clear screen is loaded.

MACHINE LANGUAGE ROUTINE

```
      LDX #0
    M LDA $D000,X
      LDY $1C00,X
      STA $1C00,X
      TYA
      STA $D000,X
      LDA $D100,X
      LDY $1D00,X
      STA $1D00,X
      TYA
      STA $D100,X
      LDA $D200,X
      LDY $1E00,X
      STA $1E00,X
      TYA
      STA $D200,X
      LDA $D300,X
      LDY $1F00,X
      STA $1F00,X
      TYA
      STA $D300,X
      DEX
      BNE M
      RTS
```

```
10 REM PAGING
20 REM
40 REM REDEFINE MEMORY-BORDERS
50 POKE134,28
100 FORI=574TO631:READA:POKEI,A:NEXTI
110 POKE12,2
115 REM MACHINE LANGUAGE ROUTINE
120 DATA162,0,189,0,208,188,0,28,157,0,28,
152,157,0,208,189,0,209
```

```
 130 DATA188,0,29,157,0,29,152,157,0,209,18
9,0,210,188,0,30,157,0,30
 140 DATA152,157,0,210,189,0,211,188,0,31,1
57,0,31,152,157,0,211
 150 DATA202,208,201,96
 155 REM ROUTINE FOR CLEAR SCREEN
 160 X=65036:Y=547:FORI=0TO25:Z=PEEK(X+I):P
OKEY+I,Z:NEXTI
 170 POKE573,96
 190 REM 600=CLEAR SCREEN
 200 GOSUB600
 205 REM 700=SWITCH PAGES
 210 GOSUB700
 220 GOSUB600
 230 PRINT"FIRST PAGE"
 235 PRINT:PRINT"1 1 1 1 1 1 1 1 1 1 1"
 240 GOSUB700
 250 PRINT"SECOND PAGE"
 255 PRINT:PRINT"2 2 2 2 2 2 2 2 2 2 2"
 260 FOR I=1 TO 1000:NEXT I
 270 GOSUB700:GOTO260
 410 POKE 134,32:END
 600 POKE11,35:X=USR(X):RETURN
 700 POKE11,62:X=USR(X):RETURN
OK
```



# Input routine

INPUT ROUTINE

Here is an input routine that prevents the program from stopping when RETURN is pressed at an input statement.
There are two ways of using that input routine:
A) If you hit RETURN nothing happens, only if you hit CTRL-RETURN will it work like the normal RETURN.
B) If you need to answer a question which must be answered with yes or with no, respectively.
For example:
1a. Press RETURN if yes. 1b. Press N-RETURN if no.
2a. Press RETURN if no. 2b. Press Y-RETURN if yes.

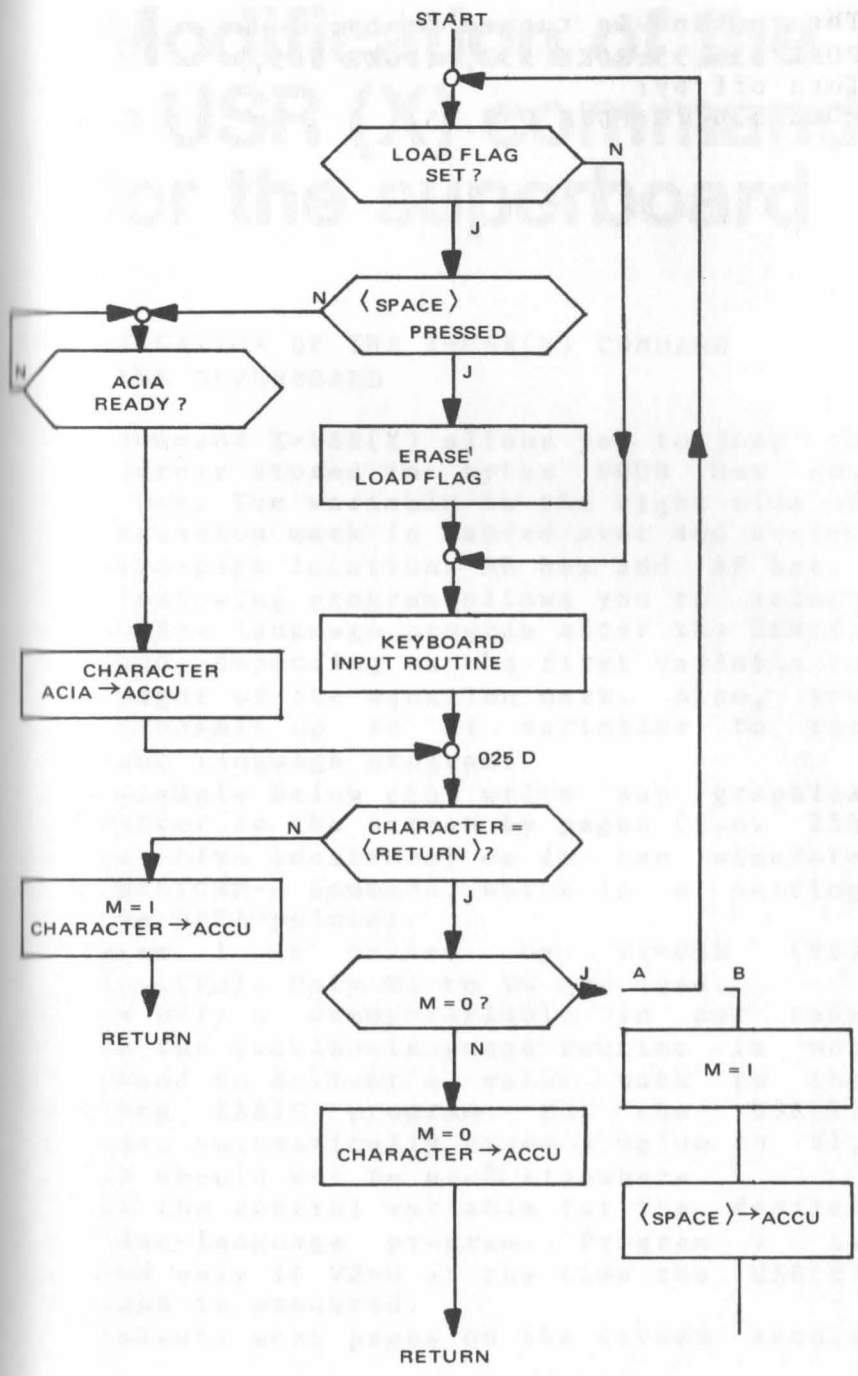
This often makes control easier. In B, you enter the ASCII character 32, which is SPACE, instead of the RETURN character. This means that you have to press RETURN twice in case 1. and 2.
Example for version B:
Question: SHOULD THE COMPUTER START (Y/N) Y ?
This means that if you answer with RETURN (2xRETURN) the question is answered with YES. (Besides that you can answer with Y and N).

Software-Routine:
The important part starts at 025D. The first part is a copy of the old input-routine which reads a character from the ACIA.

```
0218  37
0219  02

0235  00
0236
0237  2C 03 02        BIT $0203
023A  10 1E           BPL M3
023C  A9 FD     M1    LDA #$FD
023E  8D 00 DF        STA $DF00
0241  A9 10           LDA #$10
0243  2C 00 DF        BIT $DF00
0246  F0 0B           BEQ M2
0248  AD 00 F0        LDA $F000
024B  4A              LSR A
024C  90 EE           BCC M1
024E  AD 01 F0        LDA $F001
024E  AD 01 F0        LDA $F001
0251  B0 0A           BCS M4
0253  A9 00     M2    LDA #0
0255  85 FB           STA $FB
0257  8D 03 02        STA $0203
025A  20 00 FD  M3    JSR $FD00
025D  8D 36 02  M4    STA $0236
0260  C9 0D           CMP #$0D
0262  F0 09           BEQ M6
0264  A9 01           LDA #1
0266  8D 35 02  M5    STA $0235
0269  AD 36 02        LDA $0236
026C  60              RTS
026D  AD 35 02  M6    LDA $0235
0270  F0 04           BEQ M7
0272  A9 00           LDA #0
0274  F0 F0           BEQ M5
0276  A9 01     M7    LDA #1
0278  8D 35 02        STA $0235
027B  A9 20           LDA #$20
027D  60              RTS
```





The routine is turned on by:
POKE 536,55:POKE 537,2:POKE 565,0
Turn off by:
POKE 536,70:POKE 537,251



# Modification of the X=USR(X) command for the superboard

MODIFICATION OF THE X=USR(X) COMMAND
FOR THE SUPERBOARD

The command X=USR(X) allows you to jump to an address stored in bytes 000B hex and 000C hex. The variable to the right side of the equation mark is handed over and stored in zero-page locations AE hex and AF hex.
The following program allows you to select a machine language program after the USR(X) command, depending on the first variable to the right of the equation mark. Also, you can transmit up to 21 variables to the machine language program.
The example below can write any graphics character to the screen by pages (i.e. 256 consecutive locations) or it can simulate the RESTORE-n command, which is a setting of the DATA-pointer.
Program 1 is called by V1=USR (V2) (V3)...(Vn). Only V1 to V4 are used.
V1 is only a dummy-variable in our case since the machine-language routine is not supposed to deliver a value back to the calling BASIC program. But the USR(X) routine automatically gives a value to V1, so V1 should not be used elsewhere.
V2 is the control variable for the desired machine-language program. Program 1 is called only if V2=0 at the time the USR(X) command is executed.
V3 selects what pages on the screen should

remain unchanged. The diagram below shows how the four pages are influenced by V3.
V4 contains the graphic-character that should be written to the selected pages.

Program 2 also is called by V1=USR (V2) (V3)... (Vn), but it only uses V2 and V3.
V2 has to be 1 to select program 2.
V3 contains the line number of the desired DATA line.

The listed machine-language program works with 8k RAM if MEMORY SIZE ? is set to 7936.
If you want to use additional routines you can write them to 1FB0-1FCF and 1FD0-1FFF and call them with V2=2, respectively V2=3.

MEMORY MAP

1F00-1F31 routine to save the variables V1-Vn
1F32-1F5F select routine; selects routine depending on V2
1F60-1F8F screen routine
1F90-1FAF RESTOREn routine
02AF low byte from V2
02AE high byte from V2
02AD low byte from V3
02AC high byte from V3
a.s.o.



```
1F00 20 05 AE A5 AF 8D AF 02 A9 AE 85 D4 20 C2 00 C9
1F10 00 F0 1E C9 3A F0 1A 20 F5 AB 20 05 AE A6 D4 A5
1F20 AF 9D 00 02 C6 D4 A6 D4 A5 AE 9D 00 02 C6 D4 D0
1F30 DB AD AF 02 C9 00 D0 04 20 60 1F 60 C9 01 D0 04
1F40 20 90 1F 60 C9 02 D0 04 20 B0 1F 60 C9 03 D0 04
1F50 20 D0 1F 60 60 00 00 00 00 00 00 00 00 00 00 00
1F60 A2 00 AD AC 02 AC AE 02 84 D4 18 66 D4 B0 03 9D
1F70 00 D0 66 D4 B0 03 9D 00 D1 66 D4 B0 03 9D 00 D2
1F80 66 D4 B0 03 9D 00 D3 E8 D0 DB 60 00 00 00 00 00
1F90 AD AE 02 85 11 AD AD 02 85 12 20 32 A4 A5 AA 18
1FA0 69 04 90 02 E6 AB 85 8F A5 AB 85 90 60 00 00 00
1FB0 60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1FC0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1FD0 60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1FE0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1FF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
 10 FORT=1TO35:PRINT:NEXT:POKE11,0:POKE12,3
1
 20 F=1:INPUT"PROGRAM NUMBER";X:PRINT
 30 IFX=0THENINPUT"PAGE SEL.,CHARACTER";Y,Z
:PRINT:GOTO50
 40 IFX=1 THENINPUT"LINE NUMBER";Y:PRINT:F=
0
 50 X=USR(X) (Y) (Z):IF F=1 GOTO 20
 60 READ A$:PRINTA$:PRINT:GOTO20
 100 DATA "LINE 100"
 101 DATA "LINE 101"
 102 DATA "LINE 102"
 103 DATA "LINE 103"
 104 DATA "LINE 104"
 105 DATA "LINE 105"
OK
```



# Textprogram for OSI-superboard

TEXTPROGRAM FOR OSI-SUPERBOARD

This machine-language program allows you to enter text into your Superboard, edit it, print it on a printer, save it on cassette, load it from cassette.

After you have entered the program you can start it with 0333G. The menu will come up. Here you can choose:

A to enter text
B to add text
C to jump to the editor
D to print on a printer
E to save text on tape
F to load text from tape

The input of text works with released SHIFT LOCK key. For capital letters, use the left shift key.

CTRL R moves cursor backwards
CTRL F moves cursor forwards
CTRL U prints preceding line
CTRL E jump to editor

In the editor you can correct or insert text.

R moves cursor backwards
F moves cursor forwards
U prints preceding line
L1 to L9 the next 1 to 9 lines are listed on the screen

X last line number is printed
N next line is printed
C one character is deleted and can be replaced by the next pressed
I up to 250 characters may be inserted; finish with CTRL-I
D delete one character; finish with CTRL-D
T jump to beginning of text
V jump to menu

The printer routine was written for the MPI 88T printer. With CTRL N you can switch to expanded printout, with CTRL O you can switch back to normal printout.
If you save text on tape, the program records from (00F7)/(00F8) to (00F9)/(00FA). "G" starts the recording.

To load text from tape you have to press the L-key twice. After the loading you must note the endaddress and write this address to 00F9 (lower byte) and 00FA (higher byte), after BREAK and M.

To start the program enter ".0333G"

Note: The program was written for a Superboard with 24 characters/line. If you want to change that, write the new value (hex) to locations 0494 and 051F.



```
     2  3  4  5  6  7  8  9  A  B  C  D  E  F  0  1

0222 A9 0D 20 2D BF 20 7A FF A9 2E 20 75 02 A5 F8 20
0232 63 02 A5 F7 20 63 02 A9 2F 20 75 02 A2 00 A1 F7
0242 20 63 02 A9 0D 20 B1 FC A9 20 20 2D BF E6 F7 D0
0252 02 E6 F8 38 A5 F9 E5 F7 A5 FA E5 F8 10 DE 4C 33
0262 03 85 FC 20 AC FE AD CC D0 20 75 02 AD CD D0 20
0272 75 02 60 20 B1 FC 20 2D BF 60 AA 9C AD A5 0A 0D
0282 0A 71 72 69 6E 74 65 72 20 6F 6E 20 21 0A 0D 0A
0292 63 68 61 72 2E 2F 6C 69 6E 65 20 3A 20 0A 0D 0A
02A2 41 3D 38 30 20 42 3D 39 36 20 43 3D 31 32 30 20
02B2 44 3D 31 33 32 01 0A 0D 0A 6D 61 72 2E 3A 20 31
02C2 2D 39 01 0D 0A 0A 72 65 63 6F 72 64 65 72 20 74
02D2 6F 20 3C 52 45 43 4F 52 44 3E 0A 0D 0A 61 66 74
02E2 2E 20 32 30 20 73 65 63 20 12 47 01 0A 0D 0A 72
02F2 65 63 6F 72 64 65 72 20 74 6F 20 3C 50 4C 41 59
0302 3E 12 20 12 20 0D 0A 0A 65 6E 64 61 64 64 72 65
0312 73 73 20 74 6F 20 46 39 2F 46 41 20 21 21 20 20
0322 20 01 A0 00 B1 F7 C9 01 F0 06 20 07 04 4C 26 03
0332 60 A9 7C 85 F7 A9 03 85 F8 20 24 03 20 00 FD C9
0342 56 30 02 E9 20 C9 41 D0 03 4C 15 04 C9 42 D0 0B
0352 A5 F9 85 F7 A5 FA 85 F8 4C 18 04 C9 43 D0 03 4C
0362 A1 04 C9 44 D0 03 4C 3A 06 C9 45 D0 03 4C 18 06
0372 C9 46 D0 03 4C 29 06 4C 33 03 0A 0D 0A 61 3D 20
0382 69 6E 70 75 74 20 20 0A 0D 0A 62 3D 20 61 64 64
```

```
0392  20 74 65 78 74 20 20 20 20 20 0A 0D 0A 63 3D 20
03A2  65 64 69 74 6F 72 0A 0D 0A 64 3D 20 70 72 69 6E
03B2  74 0A 0D 0A 65 3D 20 73 61 76 65 20 20 20 20 20
03C2  20 20 0A 0D 0A 66 3D 20 6C 6F 61 64 20 20 20 20
03D2  20 20 20 01 A6 F7 86 F9 A6 F8 86 FA 60 C6 F7 A6
03E2  F7 E0 FF D0 02 C6 F8 60 A2 20 86 F7 A2 07 86 F8
03F2  60 A6 F7 E4 F9 D0 06 A6 F8 E4 FA F0 02 18 60 38
0402  60 A0 00 91 F7 20 2D BF E6 F7 A6 EE E0 00 D0 02
0412  E6 F8 60 20 EA 03 D8 A9 0D 20 03 04 20 00 FD C9
0422  05 D0 03 4C A1 04 20 31 04 20 D6 03 4C 1E 04 C9
0432  12 D0 06 20 5D 04 4C 1E 04 C9 06 D0 06 20 6F 04
0442  4C 1E 04 C9 15 D0 06 20 7D 04 4C 1E 04 C9 09 D0
0452  03 4C F2 05 20 03 04 60 EA EA EA 20 DF 03 CE 00
0462  02 CE 00 02 20 DF 03 B1 F7 20 03 04 60 20 F3 03
0472  90 01 60 A0 00 B1 F7 20 07 04 60 A0 30 20 DF 03
0482  88 D0 FA A0 00 B1 F7 C9 20 F0 06 20 0A 04 4C 87
0492  04 A0 16 A2 00 A1 F7 20 07 04 88 D0 F6 60 60 20
04A2  EA 03 A9 0A 20 2D BF 20 00 FD C9 5C 30 02 E9 20
04B2  C9 52 D0 06 20 5D 04 4C A9 04 C9 46 D0 06 20 6F
04C2  04 4C A9 04 C9 43 D0 03 4C 67 05 C9 44 D0 03 4C
04D2  70 05 C9 49 D0 03 4C AD 05 C9 55 D0 06 20 7D 04
04E2  4C A9 04 C9 54 D0 06 20 EA 03 4C A9 04 C9 4C D0
04F2  03 4C 53 05 C9 58 D0 07 A6 F2 86 F3 4C 0C 05 C9
0502  56 D0 03 4C 33 03 A9 01 85 F3 A9 0A 20 2D BF A9
0512  0D 20 2D BF A9 0A 20 2D BF 20 2D BF A0 18 B1 F7
0522  C9 20 F0 04 88 4C 20 05 A2 00 A1 F7 20 07 04 20
```

```
0532  F3 03 90 03 4C 5F 05 C9 0A D0 03 20 2D BF 88 F0
0542  03 4C 2A 05 A6 F3 CA F0 05 86 F3 4C 11 05 4C A9
0552  04 20 00 FD E9 2F 85 F2 85 F3 4C 0C 05 A9 F1 20
0562  2D BF 4C A9 04 20 00 FD 20 03 04 4C A9 04 A2 01
0572  86 F0 A0 01 B1 F7 20 03 04 20 00 FD C9 04 F0 05
0582  E6 F0 4C 74 05 A4 F0 20 DF 03 88 D0 FA A4 F0 C6
0592  F9 A6 F9 E0 FF D0 02 C6 FA 88 D0 F3 A4 F0 B1 F7
05A2  20 03 04 20 F3 03 90 F4 4C A9 04 A5 FA 85 F4 A5
05B2  F8 85 F6 18 A9 FF 65 F9 85 F3 90 02 E6 F4 18 A9
05C2  FF 65 F7 85 F5 90 02 E6 F6 A0 FF A2 00 A1 F9 91
05D2  F9 20 F3 03 B0 0D C6 F9 A6 F9 E0 FF D0 02 C6 FA
05E2  4C CB 05 20 00 FD C9 09 F0 06 20 31 04 4C E5 05
05F2  A0 00 B1 F5 20 05 04 A6 F5 E4 F3 D0 06 A6 F6 E4
0602  F4 F0 0D E6 F5 A6 F5 E0 00 D0 02 E6 F6 4C F4 05
0612  20 D6 03 4C A9 04 A9 C5 85 F7 A9 02 85 F8 20 24
0622  03 20 12 07 4C 22 02 A9 EE 85 F7 A9 02 85 F8 20
0632  24 03 20 00 FD 4C 00 FE A9 80 85 F7 A9 02 85 F8
0642  20 24 03 20 00 FD 20 2D BF C9 5C 30 02 E9 20 C9
0652  41 D0 08 A9 1C 85 F4 A9 4C 85 F5 C9 42 D0 08 A9
0662  1D 85 F4 A9 56 85 F5 99 43 D0 08 A9 1E 85 F4 A9
0672  73 85 F5 C9 44 D0 A9 1F 85 F4 A9 82 85 F5 A9 B8
0682  85 F7 A9 02 85 F8 20 24 03 20 00 FD C9 3C 30 02
0692  E9 20 20 2D BF E9 30 85 F3 A9 00 85 F2 A5 F4 20
06A2  B1 FC A5 F5 E5 F3 85 F6 20 EA 03 A4 F3 A9 20 20
06B2  B1 FC 88 D0 FA A4 F6 B1 F7 C9 20 F0 04 88 4C BA
```

```
06C2  06 A2 00 A1 F7 C9 0D D0 09 20 F5 06 20 FC 06 4C
06D2  AE 06 C9 0A D0 03 4C CC 06 20 F5 06 20 F3 03 90
06E2  03 4C 33 03 88 D0 DA 20 FC 06 A9 0D 20 B1 FC 4C
06F2  AE 06 20 B1 FC 20 07 04 60 E6 F2 A2 3F E4 F2 F0
0702  03 60 EA EA A2 08 A9 0A 20 B1 FC CA D0 FA 60 20
0712  EA 03 20 00 FD 60 A0 BA CA 2A E2 EA AA AA
..
```



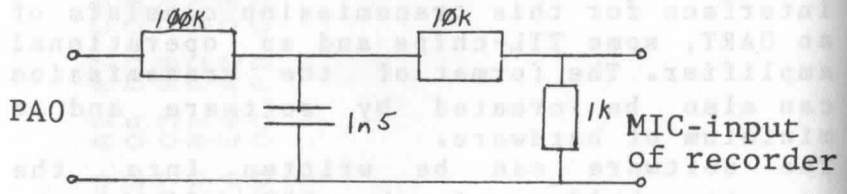
# Kansas-city interface for the PET

KANSAS-CITY INTERFACE FOR THE PET

In this chapter we describe how to create an output on the PET in the KANSAS-CITY format. A tape recorded in that format later can be read by a Superboard using the L-command of the monitor.
The KANSAS-CITY-format uses eight sine-waves of 2400 Hz to form a logical "1" and four sine-waves of 1200 Hz to form a logical "0". The transmission speed is 300 bit/sec. In the OHIO-computers the interface for this transmission consists of an UART, some TTL-chips and an operational amplifier. The format of the transmission can also be created by software and a minimium of hardware.
The software can be written into the cassette buffer of the PET. First the interrupt of the PET is disabled and PA0 is defined as an output. Then a leader of 14 seconds HI-level is created. HI is the stop-polarity on serial interfaces. Next a "/" is created to tell the OSI monitor that datas (and not addresses) are being sent. The address where the datas should go has to be entered with the OSI monitor before loading.
Then the data is split into two hex-numbers and sent. A $0D is added after each byte.
The pointer in $50,$51 for the start address and $52,$53 for the end address of

the block to be transfered have to be entered by hand before the start of the program. At the end of the input the reading has to be stopped by the break-key. The subroutine that sends an ASCII sign is called OUT 14. First a startbit is created, then eight data-bits and finally two stopbits. The subroutine HI creates eight squarewaves and the subroutine LO creates four squarewaves.
At the end a note for the PET user:
Probably you can record in KANSAS-CITY format with the installed recorder if you use the internal port at $E840 bit 3 instead of PA0. With appropriate programs you also can create the formats of APPLE, AIM, and KIM.

## Hardware of the interface:






## Machine language routine:

```
 10 0000                ,SOFT KANSAS CITY
 20 0000                ,FOR PET
 30 0000                ,TOGGLES PA0
 40 0000                B=829
 50 033D 78             SEI
 60 033E A901           LDA E$1
 70 0340 8D43E8         STA 59459
 80 0343                PAD=59457
 90 0343                ,
100 0343                ,LEADER
110 0343                ,
120 0343 20DA03         JSR VOR
130 0346 A92F           LDA E$2F
140 0348 208303         JSR OUT14
150 034B A150    START  LDA ($50,X)
160 034D 206B03         JSR OUTBYT
170 0350 A90D           LDA E$D
180 0352 208303         JSR OUT14
190 0355 E650           INC $50
200 0357 D002           BNE CONT11
210 0359 E651           INC $51
220 035B A552    CONT11 LDA $52
230 035D C550           CMP $50
240 035F D0EA           BNE START
250 0361 A553           LDA $53
260 0363 C551           CMP $51
270 0365 D0E4           BNE START
280 0367 20DA03         JSR VOR
290 036A 60             RTS ,END OF PROGRAM
300 036B                ,
310 036B                ,SUBROUTINES FOLLOW
320 036B                ,
330 036B 48      OUTBYT PHA
340 036C 4A             LSR A
350 036D 4A             LSR A
360 036E 4A             LSR A
370 036F 4A             LSR A
380 0370 207A03         JSR OUTCH
390 0373 68             PLA
400 0374 290F           AND E$F
410 0376 207A03         JSR OUTCH
420 0379 60             RTS
430 037A C90A    OUTCH  CMP E$A
440 037C 3003           BMI OUT12
450 037E 18             CLC
460 037F 6907           ADC E7
470 0381 6930    OUT12  ADC E$30
480 0383 20B403  OUT14  JSR LO
490 0386 A008           LDY E8
500 0388 4A      OUTCRY LSR A
510 0389 B006           BCS OUTHI
520 038B 20B403         JSR LO
530 038E 18             CLC
540 038F 9003           BCC CONT12
550 0391 209E03  OUTHI  JSR HI
560 0394 88      CONT12 DEY
570 0395 D0F1           BNE OUTCRY
580 0397 209E03         JSR HI
590 039A 209E03         JSR HI
```

```
 600 039D 60          RTS
 610 039E 48     HI   PHA
 620 039F A208        LDX £8
 630 03A1 A901   HI1  LDA £1
 640 03A3 8D41E8      STA PAD
 650 03A6 20D003      JSR DELY20
 660 03A9 EE41E8      INC PAD
 670 03AC 20D003      JSR DELY20
 680 03AF CA          DEX
 690 03B0 D0EF        BNE HI1
 700 03B2 68          PLA
 710 03B3 60          RTS
 720 03B4 48     LO   PHA
 730 03B5 A204        LDX £4
 740 03B7 A901   LO1  LDA £1
 750 03B9 8D41E8      STA PAD
 760 03BC 20D003      JSR DELY20
 770 03BF 20D003      JSR DELY20
 780 03C2 EE41E8      INC PAD
 790 03C5 20D003      JSR DELY20
 800 03C8 20D003      JSR DELY20
 810 03CB CA          DEX
 820 03CC D0E9        BNE LO1
 830 03CE 68          PLA
 840 03CF 60          RTS
 850 03D0 48     DELY20 PHA    ; DELAY 20 µs
 860 03D1 8A          TXA
 870 03D2 A223        LDX £35
 880 03D4 CA     LDEL DEX
 890 03D5 D0FD        BNE LDEL
 900 03D7 AA          TAX
 910 03D8 68          PLA
 920 03D9 60          RTS
 930 03DA A9F0   VOR  LDA £$F0  ; LEADER
 940 03DC A000   VOR1 LDY £0
 950 03DE 209E03 VOR2 JSR HI
 960 03E1 88          DEY
 970 03E2 D0FA        BNE VOR2
 980 03E4 18          CLC
 990 03E5 6901        ADC £1
1000 03E7 D0F3        BNE VOR1
1010 03E9 60          RTS
```



# Calculation of time in BASIC

CALCULATION OF TIME IN BASIC

The following program allows you to calculate the number of days between two dates or to calculate your own age in minutes. The program calculates the period in minutes, hours, and days. The results are rounded to whole numbers. The minutes are calculated as the minutes minus one. Therefore, the program calculates zero minutes for the difference between 12.15 and 12.16 of the same day.
The program considers all leap-years. The program uses the Gregorian calendar and assumes it was always valid. After the output you can change one or both dates. The program is written in MORD-BASIC which means that there were no special commands used so that it should work on any BASIC.

```
 10 REM TIME CALCULATION
 20 FOR I=1 TO 25:PRINT:NEXT I
 30 DIM M(12)
 40 FOR I=1 TO 12:READ M(I):NEXT I
 55 DATA -1,30,58,89,119,150,180,211,242,27
2,303,333
 60 T=0
 110 PRINT"CALCULATION OF TIME ":PRINT
 120 PRINT"BETWEEN TWO MOMENTS":PRINT
 125 PRINT:PRINT
 135 IF F1=1 THEN 210
 140 PRINT"1ST MOMENT:":PRINT
```

```
 150 INPUT"MONTH";M1:PRINT
 160 INPUT"DAY";T1:PRINT
 170 INPUT"YEAR";J1:PRINT
 180 INPUT"HOUR";S1:PRINT
 190 INPUT"MINUTES";I1:PRINT:PRINT
 195 IF F2=1 THEN 300
 210 PRINT"2ND MOMENT:":PRINT
 220 INPUT"MONTH";M2:PRINT
 230 INPUT"DAY";T2:PRINT
 240 INPUT"YEAR";J2:PRINT
 250 INPUT"HOUR";S2:PRINT
 260 INPUT"MINUTES";I2:PRINT
 300 T1=INT(ABS(T1)):T2=INT(ABS(T2)):M1=INT
(ABS(M1)):M2=INT(ABS(M2))
 310 J1=INT(ABS(J1)):J2=INT(ABS(J2)):S1=INT
(ABS(S1)):S2=INT(ABS(S2))
 320 I1=INT(ABS(I1)):I2=INT(ABS(I2))
 330 IF J2>J1 THEN 400
 340 IF J2<J1 THEN 900
 350 IF M2>M1 THEN 1100
 360 IF M2<M1 THEN 900
 370 IF T1>T2 THEN 1100
 375 IF T2<T1 THEN 900
 380 IF S2>S1 THEN S=S2-S1-1:GOTO 560
 385 IF S2<S1 THEN 900
 390 IF I2<I1 THEN 900
 395 IF I1=I2 THEN 600
 397 MI=I2-I1-1:GOTO 610
 400 IF J1>J2-2 THEN 470
 405 FOR I=J1+1 TO J2-1
 410 IF I/4<>INT(I/4) THEN 450
 430 IF (I/100=INT(I/100))AND (I/400<>INT(I
/400)) THEN 450
 440 T=T+1
 450 T=T+365
 460 NEXT I
 470 T=T-M(M1)-T1+364
 480 IF (J1/4<>INT(J1/4)) OR (M1>2) THEN 51
0
 490 IF (J1/100=INT(J1/100)) AND (J1/400<>I
NT(J1/400)) THEN 510
 500 T=T+1
 510 T=T+M(M2)+T2
```



```
 515 IF M2<3 THEN 550
 520 IF J2/4 <> INT(J2/4) THEN 550
 530 IF (J2/100=INT(J2/100)) AND (J2/400<>I
NT(J2/400)) THEN 550
 540 T=T+1
 550 S=T*24+23-S1+S2
 560 MI=S*60+59-I1+I2
 570 GOTO 610
 600 MI=0
 610 PRINT"# OF DAYS";INT(MI/1440+.5):PRINT
 620 PRINT"# OF HOURS";INT(MI/60+.5):PRINT
 630 PRINT"# OF MINUTES";INT(MI+.5):PRINT:P
RINT
 640 PRINT"CHANGE 1ST MOMENT";:GOSUB 800:F1
=I
 650 PRINT"CHANGE 2ND MOMENT";:GOSUB 800:F2
=I
 700 IF (F1=0) OR (F2=0) THEN 60
 710 PRINT:PRINT"GOOD BYE !":END
 800 INPUT"Y/N";A$:PRINT:IF A$="Y" THEN I=0
:GOTO 810
 805 IF A$="N" THEN I=1:GOTO 810
 807 GOTO 800
 810 RETURN
 900 PRINT:PRINT"2ND MOMENT HAS TO BE":PRIN
T
 910 PRINT"AFTER THE 1ST":PRINT:PRINT:GOTO
60
 1100 T=M(M2)+T2-M(M1)-T1-1
 1110 IF ((M2>2) AND (M1>2)) OR ((M2<3) AND
 M1<3)) THEN 550
 1130 GOTO 520
OK
```

# Dynic

DYNIC

This game is played on a board with 8x8 squares. You play against the computer. The first stone has to be placed in one of the center squares (see figure 1). Then the computer places a stone and so on. The stones always have to be placed in the neighborhood of another stone (see figure 2). Try to surround the other stones and make them yours (see figure 3). Stones that are surrounded diagonally will not be changed (see figure 4). You play till all squares are occupied. The one with more stones wins.

Figure 1

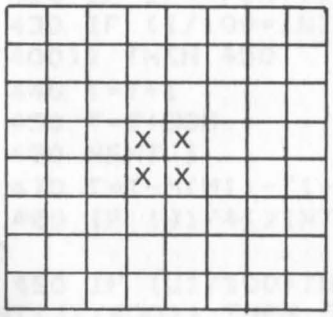


Figure 2

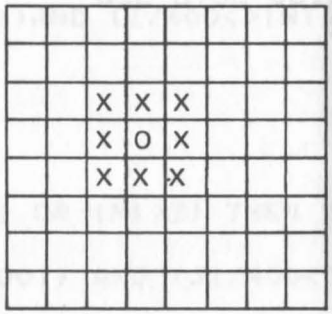




Figure 3

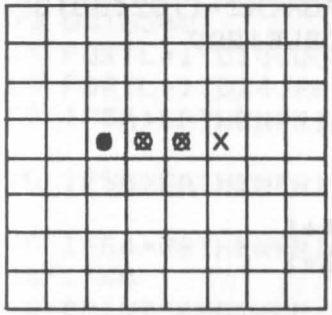
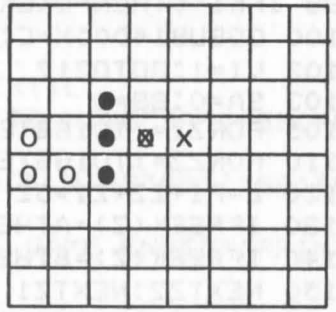


Figure 4



```
10 CLEAR
11 REM START WITH D4,E4
12 REM OR WITH D5,E5
13 REM PUT YOUR STONES IN
14 REM NEIGHBORHOOD OF
15 REM OTHER STONES
16 REM CHANGE OF PLAYERS
17 REM WITH ESC-KEY
18 REM SURROUNDED STONES
19 REM WILL BE CHANGED
20 REM CORRECTIONS BY RUBOUT
21 F1=53443+4:F2=F1-1
23 F5=53411+9
25 F3=53987+4:F4=F3-33
30 FORTL=1TO32:PRINT:NEXTTL
31 GOSUB4000
33 NZ$="NEXT DRAW: "
35 FE$="WRONG INPUT"
37 A=232:B=233
40 GOSUB1000
45 AA=54116:K2=54132
50 CO=11:K=57088
55 DIME(65)
56 DIMZG(50)
60 C=A:D=B
90 FORT=1TOCO:POKEAA+T,ASC(MID$(NZ$,T,1)):N
EXTT
95 GOTO1100
96 GOSUB1370
97 GOSUB1270
```

```
98 GOSUB1380:GOSUB1230
99 IFK1=0THENPOKEK2,32:POKEK2+1,32:GOTO95
100 GOSUB1400:X=C:Y=D:GOSUB1800
102 VI=1:GOTO217
103 SA=0:SB=0
105 FORZ1=1TO16STEP2
110 FORZ2=1TO16STEP2
120 Z=F1+Z2+Z1*32
130 IFPEEK(Z)=ATHENSA=SA+1
140 IFPEEK(Z)=BTHENSB=SB+1
150 NEXTZ2:NEXTZ1
160 S1=48+INT(SA/10)
170 S2=48+10*(SA/10-INT(SA/10))
180 S3=48+INT(SB/10)
190 S4=48+10*(SB/10-INT(SB/10))
200 POKEF5,S1:POKEF5+1,S2:POKEF5+2,58
210 POKEF5+3,S3:POKEF5+4,S4
215 POKEK2,32:POKEK2+1,32
216 RETURN
217 GOSUB103
220 I=0
230 FORZ1=1TO16STEP2
240 FORZ2=1TO16STEP2
250 ZUG=F1+Z2+32*Z1
260 GOSUB1460
270 IFPEEK(ZU)=32THEN1600
280 IFPEEK(ZU)=96THENPOKEZU,32
290 NEXTZ2:NEXTZ1
295 I2=0:I3=0
300 FORI1=1TOI
310 IFI2<E(I1)THENI2=E(I1):I3=I1
320 NEXTI1
325 IFI3=0THENI3=1
330 POKEZG(I3),D
335 GOTO360
340 FORI1=1TOI
350 ZG(I1)=0:NEXTI1
355 GOTO95
360 X=D
370 Y=C
375 ZU=ZG(I3)
450 GOSUB1800
460 GOSUB103
```



```
480 IFSA+SB=64THEN505
500 GOTO340
505 FORTL=1TO10000:NEXTTL
510 FORTL=1TO24:PRINT:NEXTTL
520 IFSA>SBTHENPRINT"WHITE WINS"SA":"SB:PRI
NT
530 IFSB>SATHENPRINT"BLACK WINS"SB":"SA:PRI
NT
540 IFSA=SBTHENPRINT"THE GAME ENDED":PRINT"
"SA":"SB
550 PRINT"ANOTHER GAME";
560 INPUTXY$
570 IFLEFT$(XY$,1)="Y"THENGOTO10
580 GOTO9999
1000 FORZ1=1TO16STEP2
1010 FORZ2=1TO16STEP2
1020 Z=F2+Z2+32*Z1
1025 POKEZ+32,219
1030 POKEZ,149
1035 POKEZ+33,148
1040 NEXTZ2:NEXTZ1
1050 Z3=65:Z4=49
1060 FORZ1=1TO16STEP2
1070 POKEF3+Z1,Z3:POKEF3+Z1-1,149:Z3=Z3+1
1080 POKEF4-Z1*32,Z4:POKEF4-Z1*32+32,148:Z4
=Z4+1
1090 NEXT
1095 RETURN
1100 POKE530,1
1103 POKEK,254
1107 IFPEEK(K)=222THENPOKE530,0:GOTO3000
1110 POKEK,253
1120 IFPEEK(K)=191THENK1=65:POKE530,0:GOTO9
6
1130 POKEK,251
1140 IFPEEK(K)=191THENK1=67:POKE530,0:GOTO9
6
1150 IFPEEK(K)=239THENK1=66:POKE530,0:GOTO9
6
1160 POKEK,247
1170 IFPEEK(K)=191THENK1=68:POKE530,0:GOTO9
6
1180 IFPEEK(K)=223THENK1=70:POKE530,0:GOTO9
```

```
6
 1190 IFPEEK(K)=239THENK1=71:POKE530,0:GOTO9
6
 1200 IFPEEK(K)=247THENK1=72:POKE530,0:GOTO9
6
 1210 POKEK,239
 1220 IFPEEK(K)=191THENK1=69:POKE530,0:GOTO9
6
 1225 GOTO1100
 1230 POKE530,1:POKEK,223
 1240 IFPEEK(K)=247THENK1=1:POKE530,0:RETURN
 1250 POKEK,191
 1255 IFPEEK(K)=251THENK1=0:POKE530,0:RETURN
 1265 GOTO1230
 1270 POKE530,1:POKEK,191
 1275 IFPEEK(K)=127THENK1=56:POKE530,0:RETUR
N
 1280 POKEK,127
 1290 IFPEEK(K)=127THENK1=49:POKE530,0:RETUR
N
 1300 IFPEEK(K)=191THENK1=50:POKE530,0:RETUR
N
 1310 IFPEEK(K)=223THENK1=51:POKE530,0:RETUR
N
 1320 IFPEEK(K)=239THENK1=52:POKE530,0:RETUR
N
 1330 IFPEEK(K)=247THENK1=53:POKE530,0:RETUR
N
 1340 IFPEEK(K)=251THENK1=54:POKE530,0:RETUR
N
 1350 IFPEEK(K)=253THENK1=55:POKE530,0:RETUR
N
 1360 GOTO1270
 1370 POKEK2,K1:RETURN
 1380 POKEK2+1,K1
 1390 FORTL=1TO200:NEXTTL:RETURN
 1400 ZUG=F2+32+2*(PEEK(K2)-64)+64*(7-PEEK(K
2+1)+49)
 1403 GOSUB1460
 1405 IFPEEK(ZUG)<>32THENGOSUB1430
 1410 POKEZUG,C
 1415 Z1=2*(7-PEEK(K2+1)+49):Z2=2*(PEEK(K2)-
64)
```



```
 1420 RETURN
 1430 FORT=1TOCO:POKEAA+T,ASC(MID$(FE$,T,1))
 1435 NEXTT
 1437 POKEK2,32:POKEK2+1,32
 1440 FORTL=1TO2000:NEXTTL
 1445 IFPEEK(ZU)=96THENPOKEZU,32
 1450 GOTO90
 1460 SU=0
 1470 SU=PEEK(ZU+2)+PEEK(ZU-2)+PEEK(ZU+64)+P
EEK(ZU-64)+PEEK(ZU+66)
 1480 SU=SU+PEEK(ZU-66)+PEEK(ZU+62)+PEEK(ZU-
62)
 1485 IFVI=0GOTO1510
 1487 IFPEEK(ZU)<>32GOTO1500
 1490 IFSU<400THENPOKEZU,96
 1500 RETURN
 1510 IFZU=53667+110RZU=53667+130RZU=53731+1
10RZU=53731+13THEN1500
 1520 POKEZU,96
 1530 GOTO1500
 1600 POKEZU,D
 1610 I=I+1
 1620 E(I)=3
 1630 GOTO1690
 1640 J=0:I1=0
 1650 I1=I1+1:J=J+J1
 1660 IFPEEK(ZU+J)=CTHEN1650
 1665 IFJ=J1THEN1680
 1670 IFPEEK(ZU+J)=DTHENE(I)=E(I)+I1
 1680 RETURN
 1690 J1=2
 1695 I2=1
 1700 GOSUB1640
 1705 I2=2
 1710 J1=-2
 1720 GOSUB1640
 1725 I2=3
 1730 J1=64
 1740 GOSUB1640
 1745 I2=4
 1750 J1=-64
 1760 GOSUB1640
 1765 GOSUB2200
```

```
1770 POKEZU,32
1780 GOTO290
1800 I=0
1810 I=I+1
1820 J1=2
1825 I2=1
1830 GOSUB1900
1840 J1=-2
1845 I2=2
1850 GOSUB1900
1860 J1=64
1865 I2=3
1870 GOSUB1900
1880 J1=-64
1883 I2=4
1885 GOSUB1900
1890 RETURN
1900 J=0:I1=0
1910 I1=I1+1:J=J+J1
1940 IFPEEK(ZU+J)=YTHEN1910
1950 IFPEEK(ZU+J)=XTHEN2100
1960 RETURN
2100 FORQZ=ZU+J1TOZU+J-J1STEPJ1
2110 POKEQZ,X
2120 NEXTQZ
2130 GOTO1960
2200 FORP=1TO4
2210 READE1(P)
2220 IFE1(P)=ZUTHENE(I)=E(I)+4
2230 NEXTP
2250 FORP=1TO4
2260 FORPA=1TO3
2270 READKO
2290 IFKO=ZUTHENE(I)=E(I)-3
2300 IFKO=ZUANDPEEK(E1(P))=DTHENE(I)=E(I)+7
2305 NEXTPA:NEXTP
2310 RESTORE
2380 ZG(I)=ZU
2390 RETURN
2400 REM TEXT
2430 I$="CHANGE OF STONES"
2440 FORT=1TOLEN(I$)
2450 POKE54084+T,ASC(MID$(I$,T,1)):NEXTT
```



```
2490 FORTL=1TO2000:NEXTTL
2500 FORT=1TOLEN(I$)
2510 POKE54084+T,ASC(" "):NEXTT
2520 RETURN
2560 GOTO2550
2570 K8=1
2575 K2=K2+4
2576 POKEK2-1,58
2580 GOTO95
2590 POKEK2-1,32:K2=K2+4:POKEK2,32:POKEK2+1
,32:GOTO2550
2960 K8=0
2970 FORT=1TO55:POKE54084+T,32:NEXTT
2980 FORT=1TOCO:POKEAA+T,ASC(MID$(NZ$,T,1))
:NEXTT
2990 POKE530,1:RETURN
3000 GOSUB2400
3010 IFC=ATHENC=B:D=A:GOTO102
3020 IFC=BTHENC=A:D=B:GOTO102
4000 PRINT"DDD  Y   Y N   N I  CCC"
4010 PRINT"D  D  Y Y  NN  N I C"
4020 PRINT"D  D   Y   N N N I C"
4030 PRINT"D  D   Y   N  NN I C"
4040 PRINT"DDD    Y   N   N I  CCC"
4050 PRINT:PRINT:PRINT:PRINT:PRINT
4060 FORTL=1TO3E3:NEXT
4900 FORL=1TO32:PRINT:NEXT
5000 POKE53417,87
5010 POKE53427,83
5100 RETURN
8000 DATA53480,53494,53928,53942
9000 DATA53482,53544,53546,53492,53556,5355
8,53864,53866,53930,53876
9001 DATA53878,53940
9999 END
OK
```

# Biorhythm

```
 100 REM BIORHYTHM
 110 REM
 120 FORQ=1TO30:PRINT:NEXTQ
 130 PI=3.14159 : K=2*PI : : DIM F(12),J(2),
O$(55)
 140 F(1)=31:F(2)=28:F(3)=31: F(4)=30: F(5)=
31: F(6)=30
 150 F(7)=31:F(8)=31:F(9)=30:F(10)=31:F(11)=
30:F(12)=31
 155 REM
 156 REM INPUT OF DATAS
 157 REM
 160 INPUT"NAME ";Z$
 170 INPUT"DAY OF BIRTH (MONTH,DAY,YEAR) ";M
1,D1,Y1
 180 IF D2>31 OR M2>12 OR Y2>99 THEN 170
 190 D2=D1 : M2=M1 : Y2=Y1 : GOSUB 1000 (SUM
DAY) : S1=S
 200 INPUT"START OF ANALYSIS (MONTH,DAY,YEAR
) ";M2,D2,Y2
 210 IF D2>31 OR M2>12 OR Y2>99 THEN 200
 220 GOSUB 1000 (SUMDAY) : S2=S : O=S2-S1
 230 INPUT"NUMBER OF DAYS";L
 240 INPUT"ALL GRAPHS (Y/N) ";A$ : IF LEFT$(
A$,1)="N" THEN 260
 250 PP=1 : EE=1 : II=1 : AA=1 : GOTO 500
 260 INPUT"INTELLIGENCE";A$ : IF LEFT$(A$,1)
<>"N" THEN II=1
 270 INPUT"CONSTITUTION";A$ : IF LEFT$(A$,1)
<>"N" THEN PP=1
 280 INPUT"FEELING  ";A$ : IF LEFT$(A$,1)<>"
N" THEN EE=1
```



```
 290 INPUT"TOTAL      ";A$ : IF LEFT$(A$,1)<>
"N" THEN AA=1
 400 REM
 500 REM OUTPUT OF RESULTS
 505 REM
 510 FORI=1TO5:PRINT" ----- ";:NEXT
 520 PRINT:PRINT"BIORHYTHM FOR    "Z$:PRINT
 530 GT$=RIGHT$(STR$(D1),2)+"."+RIGHT$(STR$(
M1),2)+"."+STR$(Y1)
 540 PRINT"DAY OF BIRTH          "GT$:PRINT
 550 GT$=RIGHT$(STR$(D2),2)+"."+RIGHT$(STR$(
M2),2)+"."+STR$(Y2)
 560 PRINT"START OF CALCULATION   "GT$:PRINT
 570 PRINT:PRINT"C - CONSTITUTION
 575 PRINT:PRINT"F - FEELING
 580 PRINT:PRINT"I - INTELLIGENCE
 585 PRINT:PRINT"T - TOTAL
 589 PRINT:PRINT
 590 PRINT"DAY      NEGATIV                CRITIC
AL       POSITIV"
 595 PRINT
 600 REM
 610 REM START OF LOOP O
 620 REM
 630 FOR T=0 TO 55 : O$(T)=" " : NEXT T
 640 L=O+L : C=O
 650 IF Y2/4-INT(Y2/4)=0 THEN F(2)=29
 660 FOR O=O TO L-1 : C=C+1 : Y=O : O$(25)="
*"
 670 REM
 680 REM CALCULATION OF THE DATA (I,C,F,T)
 690 REM
 700 X=INT((SIN(K*(O/23-INT(O/23)))*25)+26)
 710 Y=Y+X : IF PP=1 THEN O$(X)="C"
 720 X=INT((SIN(K*(O/33-INT(O/33)))*25)+26)
 730 Y=Y+X : IF II=1 THEN O$(X)="I"
 740 X=INT((SIN(K*(O/28-INT(O/28)))*25)+26)
 750 Y=(Y+X)/3 : IF EE=1 THEN O$(X)="F"
 760 IF AA=1 THEN O$(Y)="T"
 770 REM
 780 REM PRINT THE DATA AT DAY O
 790 REM
 800 IF D2=1 THEN PRINT : PRINT"  "M2;Y2+190
```

```
O : PRINT
 810 PRINT D2;TAB(5);" ";:
 820 FOR I=1 TO 55 : PRINTO$(I); : O$(I)=" "
 : NEXT I : PRINT
 830 D2=D2+1 : IF D2<=F(M2) THEN 850
 840 D2=1 : M2=M2+1
 850 IF M2<13 THEN 870
 860 D2=1 : M2=1 : Y2=Y2+1
 870 IF Y2/4-INT(Y2/4)<>0 THEN 890
 880 F(2)=29 : GOTO 900
 890 F(2)=28
 900 NEXT O
 910 REM
 920 REM END OF LOOP O
 930 REM
 940 CLEAR : GOTO 156
 1000 REM
 1010 REM --------------------
 1020 REM SUBROUTINE SUMDAY
 1030 REM DAY+MONTH+YEAR (D2,M2,Y2) ---> S
 1040 REM
 1050 Z=0 : IF Y2/4<>INT(Y2/4) THEN 1080
 1060 IF M2<=2 THEN 1080
 1070 Z=1
 1080 Z=Z+(Y2-1)*365+INT((Y2-1)/4)
 1090 X=M2 : GOSUB 1100 (DAYS) : S=Z+J2+D2 :
 RETURN
 1100 REM
 1110 REM ------------------
 1120 REM SUBROUTINE DAYS
 1130 REM DAYS OF MONTHS 1. TO X-1. ---> J2
 1140 REM
 1150 J2=0 : IF X=1 THEN RETURN
 1160 FOR I=1 TO X-1 : J2=J2+F(I) : NEXTI
 1170 RETURN
OK
```



# Mailing list for Ohio scientific

MAILING LIST FOR OHIO SCIENTIFIC

After you loaded and started the program the menu will appear:

1.) START A NEW DATA DISK
2.) ENTER ADDRESSES
3.) PRINT ADDRESSES
4.) INSTRUCTIONS
5.) END OF MENU
6.) CHANGE ADDRESSES

Enter the function you need.

The addresses contain the following information:

NO. OF CUSTOMER
COMPANY
ATTN.
NO. AND STREET
CITY
STATE,ZIP-CODE
PHONE-NO.
PARAMETER 1
PARAMETER 2
PARAMETER 3

If you want to print addresses you can print single addresses or a list of addresses on labels or a phone list or a list of parameters.

The name of the data-file that holds the addresses is "SCRA".
One diskette holds up to 235 addresses.
After you have entered your addresses you can search for a certain number.
The printout of addresses can be directed to the screen or to a printer.

```
 5 DIMA$(5),B$(20),C$(15),E$(10),F$(20),G$(1
5)
 6 DIMD$(20),H$(5),I$(5),K$(5),L$(72),T$(60)
 7 DIMM$(72),N$(60),O$(10),P$(20)
 10 REM MASTERPROGRAMM
 20 GOSUB 3000
 30 FORX=1TO10:PRINT:NEXTX
 40 DEF FNA(V)=((24-LEN(AA$))/2):REM CENTER
OF SCREEN
 60 AA$="MAILING LIST"
 70 GOSUB 4000
 75 AA$="FOR"
 80 GOSUB 4000
 90 AA$="OHIO SCIENTIFIC"
 100 GOSUB 4000
 110 AA$="C1P MF"
 120 PRINT:PRINT:PRINT:
 130 GOSUB 3000
 140 AA$="MENUE"
 150 GOSUB 4000
 160 AB$="1.START A NEW DATA-DISK":GOSUB5000
 180 AB$="2.ENTER ADDRESSES":GOSUB5000
 190 AB$="3.PRINT ADDRESSES":GOSUB5000
 200 AB$="4.INSTRUCTIONS":GOSUB5000
 210 AB$="5.END OF MENU":GOSUB5000
 212 AB$="6.CHANGE":GOSUB5000
 215 PRINT:PRINT
 220 AB$="ENTER NUMBER":GOSUB5020
 240 INPUT BB
 250 IFBB>6ORBB<1THEN240
 260 IFBB=1 THEN GOSUB 3500
 270 IFBB=2THENGOSUB300
 280 IF BB=3 THEN GOSUB 7000
 285 IFBB=4THENGOSUB6000
```



```
290 IFBB=5THEN GOTO10
293 IFBB=6THENGOSUB6500
294 GOSUB530
295 GOTO 130
300 REM LIST OF ADDRSSES
330 INPUT"NO. OF CUSTOMER";A$
340 IF VAL(A$)=0THEN 10
350 INPUT"COMPANY: ";B$
360 PRINT:INPUT"ATTN.: ";C$
370 INPUT"NO. AND STREET: ";D$
380 INPUT"CITY: ";E$
390 INPUT"STATE,ZIP-CODE: ";F$
400 INPUT"PHONE NO.: ";G$
410 INPUT"PARAMETER 1: ";H$
412 INPUT"PARAMETER 2: ";I$
414 INPUT"PARAMETER 3: ";K$
415 PRINT:PRINT:PRINT
417 GOSUB 3000
420 PRINT"THE INPUT IS FINISHED"
425 PRINT"DO YOU WANT TO CHANGE"
427 PRINT"SOMETHING ?"
430 PRINT
440 PRINT"NO. OF CUSTOMER ";A$
450 PRINT:PRINT"COMPANY: ";B$
460 PRINT:PRINT"ATTN.: ";C$
470 PRINT:PRINT"NO. AND STREET: ";D$
480 PRINT:PRINT"CITY: ";E$
490 PRINT:PRINT"STATE,ZIP-CODE: ";F$
500 PRINT:PRINT"PHONE-NO.: ";G$
510 PRINT"PARAMETER 1: ";H$
512 PRINT"PARAMETER 2: ";I$
514 PRINT"PARAMETER 3: ";K$
520 PRINT:PRINT"IS EVERYTHING OK ";
525 INPUT M$
530 IF M$<>"Y"THEN GOSUB 6520
535 GOSUB 700
537 I=VAL(A$)
540 DISK OPEN,6,"SCRA"
550 DISK GET,I-1
560 PRINT#6,L$
565 PRINT#6,T$
640 DISK PUT
650 DISK CLOSE,6
```

```
660 GOTO330
700 REM PUT STRINGS TOGETHER
710 N$="      "
720 O$="            "
730 P$="                  "
740 Q$="                        "
750 A$=LEFT$(A$+N$,5)
760 B$=LEFT$(B$+Q$,20)
770 C$=LEFT$(C$+P$,15)
780 D$=LEFT$(D$+Q$,20)
790 E$=LEFT$(E$+O$,10)
800 F$=LEFT$(F$+Q$,20)
810 G$=LEFT$(G$+P$,15)
820 H$=LEFT$(H$+N$,5)
830 I$=LEFT$(I$+N$,5)
840 K$=LEFT$(K$+N$,5)
850 L$=A$+B$+C$+D$+E$
855 T$=F$+G$+H$+I$+K$
860 RETURN
865 RETURN
1000 REM READ-PROGRAM
1005 PRINT"SEARCH FOR NUMBERS (NU)"
1007 PRINT"OR NAMES (NA)"
1010 PRINT:INPUT NA$
1020 IF NA$<>"NU" THEN 2000
1200 REM SEARCH FOR NUMBER
1210 INPUT"ENTER NO. OF CUSTOMER: ";A
1220 DISK OPEN,6,"SCRA"
1240 DISK GET,A-1
1250 INPUT#6,L$
1255 INPUT#6,T$
1270 A$=LEFT$(L$,5)
1280 B$=MID$(L$,6,20)
1290 C$=MID$(L$,26,15)
1300 D$=MID$(L$,41,20)
1310 E$=MID$(L$,61,10)
1320 F$=LEFT$(T$,20)
1330 G$=MID$(T$,21,15)
1340 H$=MID$(T$,36,5)
1350 I$=MID$(T$,41,5)
1360 K$=MID$(T$,46,5)
1370 DISK CLOSE,6
1410 REM PRINTER
```



```
1420 GOSUB 3000
1421 PRINT"SCREEN(1) OR PRINTER(2)"
1422 INPUT PP
1423 IF PP<1ORPP>2THEN 1422
1424 IF PP=2 THEN POKE 8994,3
1430 PRINT"(1) NO. OF CUSTOMER: ";A$
1440 PRINT:PRINT"(2) COMPANY: ";B$
1450 PRINT:PRINT"(3) ATTN.: ";C$
1460 PRINT:PRINT"(4) NO. AND STREET: ";D$
1470 PRINT:PRINT"(5)(6) CITY,STATE,ZIP: ";E
$;F$
1475 PRINT:PRINT"(7) PHONE NO.: ";G$
1480 PRINT:PRINT"(8)(9)(10) PARAMETER"
1490 PRINT;H$;I$;K$
1491 INPUT"ENTER C TO CONTINUE";CB$
1492 IFCB$="C"THEN1493
1493 POKE 8994,2
1495 RETURN
1500 END
2000 PRINT"NOT YET FINISHED"
3000 FOR X=1 TO 30:PRINT:NEXT X
3010 RETURN
3015 FORX=1TO 5:PRINT:NEXTX
3500 PRINT"STILL WORKING"
3510 GOTO140
4000 PRINT TAB(FNA(V));AA$
4010 RETURN
5000 REM SCREEN POSITION
5010 PRINT TAB(3);AB$:RETURN
5020 PRINT TAB(5);AB$:RETURN
5030 PRINT TAB(10);AB$:RETURN
5100 RETURN
6000 PRINT"STILL WORKING":GOTO140
6020 RETURN
6500 GOSUB 3000
6510 PRINT"WHICH CUSTOMERS NO. TO CHANGE":G
OSUB1210
6520 PRINT"WHAT TO CHANGE"
6530 GOSUB 1430
6540 PRINT"ENTER NO."
6550 INPUT CC
6560 ON CC GOSUB 6600,6610,6620,6630,6640,6
650,6660,6670,6680,6690
```

```
6570 GOTO 420
6600 INPUT"(1) NO. OF CUSTOMER ";A$:RETURN
6610 INPUT"(2) COMPANY ";B$:RETURN
6620 INPUT"(3) ATTN. ";C$:RETURN
6630 INPUT"(4) NO. AND STREET ";D$:RETURN
6640 INPUT"(5) CITY ";E$:RETURN
6650 INPUT"(6) STATE,ZIP CODE ";F$:RETURN
6660 INPUT"(7) PHONE NO. ";G$:RETURN
6670 INPUT"(8) PARAMETER1 ";H$:RETURN
6680 INPUT"(9) PARAMETER2 ";I$:RETURN
6690 INPUT"(10) PARAMETER3 ";K$:RETURN
6700 RETURN
7000 GOSUB 3000
7010 PRINT:PRINT:PRINT
7020 AB$="(1) SINGLE ADDRESSES":GOSUB5000
7030 AB$="(2) LABELS":GOSUB5000
7040 AB$="(3) LIST OF ADDRESSES":GOSUB5000
7045 AB$="(4) END":GOSUB5000
7050 PRINT:PRINT
7060 AB$="ENTER NO.":GOSUB5000
7080 INPUT BB
7090 IF BB>4ORBB<1THEN 7080
7100 IF BB=1 THEN GOSUB 1000:GOTO 130
7110 IF BB=2 THEN GOSUB 7500:GOTO 130
7120 IF BB=3 THEN GOSUB  8000:GOTO 130
7500 REM PRINT LABELS
7510 PRINT"FIRST NO."
7515 INPUT T
7517 INPUT"LAST NO.";AD
7520 INPUT"TAB POSITIONS";V,W
7540 INPUT"NO. OF LINE FEEDS";AA
7550 DISK OPEN,6,"SCRA"
7555 TT=T
7560 DISK GET,TT-1
7590 INPUT#6,L$
7600 INPUT#6,T$
7602 INPUT#6,M$
7604 INPUT#6,N$
7680 REM PRINT OUT
7685 POKE 8994,3
7700 PRINTTAB(V);LEFT$(L$,5);TAB(W);LEFT$(M
$,5)
7710 PRINTTAB(V);MID$(L$,6,20);TAB(W);MID$(
```



```
M$,6,20)
7720 PRINTTAB(V);MID$(L$,26,15);TAB(W);MID$
(M$,26,15)
7730 PRINTTAB(V);MID$(L$,41,20);TAB(W);MID$
(M$,41,20)
7740 E$=MID$(L$,61,7)
7750 F$=LEFT$(T$,20)
7760 O$=MID$(M$,61,7)
7770 P$=LEFT$(N$,20)
7780 PRINTTAB(V);E$;F$;TAB(W);O$;P$
7781 FOR E5=1 TO AA:PRINT:NEXT E5
7785 POKE 8994,2
7790 IF TT<AD THEN GOTO 7810
7800 DISK CLOSE,6:RETURN
7810 T=TT+2:GOTO7555
8000 REM OUTPUT OF ADDRESSES
8010 GOSUB 3000
8020 AB$="(1) LIST OF ADDRESSES":GOSUB5000
8030 AB$="(2) PHONE LIST":GOSUB5000
8040 AB$="(3) LIST OF PARAMETERS":GOSUB5000
8050 AB$="(4) END":GOSUB5000
8060 PRINT:PRINT:PRINT
8070 AB$="ENTER NO.":GOSUB5000
8080 INPUT BB
8085 IF BB=4 GOTO 10
8100 IF BB>4ORBB<1 THEN 8080
8110 IFBB=1ORBB=2ORBB=3 THEN GOSUB 8200
8200 REM OUTPUT OF ADDRESSES
8210 INPUT"FIRST,LAST NO.";T,AD
8220 DISK OPEN,6,"SCRA"
8230 DISK GET,T-1
8240 INPUT#6,L$
8250 INPUT#6,T$
8260 GOSUB 8400
8261 IF BB=1 THEN GOSUB 8700
8262 IF BB=2 THEN GOSUB 8710
8263 IF BB=3 THEN GOSUB 8720
8290 IF T=AD THEN DISK CLOSE,6:GOTO10
8300 T=T+1:GOTO 8230
8400 REM PUT STRINGS TOGETHER
8410 A$=LEFT$(L$,5)
8420 B$=MID$(L$,6,20)
8430 C$=MID$(L$,26,15)
```

```
8440 D$=MID$(L$,41,20)
8450 E$=MID$(L$,61,10)
8460 F$=LEFT$(T$,20)
8470 G$=MID$(T$,21,15)
8480 H$=MID$(T$,36,5)
8490 I$=MID$(T$,41,5)
8500 K$=MID$(T$,46,5)
8510 RETURN
8700 PRINT#1,A$;B$;D$;E$;F$:PRINT:RETURN
8710 PRINT#1,A$;B$;C$;G$:PRINT:RETURN
8720 PRINT#1,A$;B$;F$;H$;I$;K$:PRINT:RETURN
```




# Writing invoices with the OSI C1P, C1P MF, or C4P MF

WRITING INVOICES WITH THE OSI C1P, C1P MF, OR C4P MF

The C1P and C4P are very powerful, yet reasonably priced, microcomputers which can be used in the small business environment. A very small business could find that an inexpensive invoice-writing program would effectively save labor, time, and money.

The following invoice-writing programs allow the user to write invoices on a special form that our staff has developed for that purpose, or on plain continuous computer paper.
Three versions are available
1. INV1: Writes an invoice using the C1P with cassette tape on regular paper.
2. INV2: Writes invoices using the C1P MF/C4P MF on regular paper.
3. INV3: Writes invoices using the C1P MF/C4P MF on the special forms.

A serial printer is needed for all three programs.
It goes without saying that the cassette version, INV1, does not offer the capability and versatility found in INV2 and INV3, on disk.
To use the programs, your products must have product numbers and fixed retail prices. The discount is automatically

calculated depending upon the quantity ordered and the shipping and handling charges according to the quantity shipped. The figures for these calculations are easily modified by changing the appropriate line in BASIC in the program.
PLEASE NOTE: The product numbers, product name, and prices must be put into DATA statement prior to beginning the invoice-writing.

HARDWARE REQUIREMENTS FOR INV1

For INV1, a Superboard 11 or 111 or a C1P with 8K RAM, 8K BASIC ROM, a cassette recorder, and a serial printer connected to the ACIA port is needed. The program runs at 300 BAUD.

HARDWARE REQUIREMENTS FOR INV2 AND INV3

To run these two programs, a C1P MF or a C4P MF with a minimum of 20K RAM and one minifloppy disk drive with a serial printer connected to Device 1 is needed.
INV1
Load the cassette into RAM with the LOAD command and start the program with the RUN command. The program first asks for Sales Tax. If it is not necessary to calculate Sales Tax, type in a zero. The date and the first invoice number are entered next. The computer increments the invoice number automatically. The date has to be entered only once for each session. Both of these entries are valid until the computer is turned off. The account number is now entered. After this, the product number and quantity are entered. The program now calculates the total and asks for the next product number and quantity. This input is terminated by typing a zero instead of the product number. The discount, product



total, tax, shipping, and grand total are calculated and printed.
INV2 and INV3
These are the enhanced versions of the program. They run in the same manner, basically, as INV1; however, they offer the ability to enter more information on the invoice with greater flexibility.
As in INV1, the sales tax is requested. Then, the date and invoice number are called for. As in INV1, both of these entries are made only once during each session and are valid until the computer is turned off. At this point, some of the added capabilities are evident.
Using INV3, shipping costs can be entered manually or automatically. By entering NO to the question,the shipping charge is entered manually.
Returning to both versions, the customer order number as well as the ability to designate a different billing and shipping address plus the option of specifying the mode of transportation and payment terms are included.
The customer order number, name, and address are then input. The computer will ask if the order is to be shipped to the same address. If not, an N is typed and the shipping address can be entered. The computer then prints the date, account number, invoice number, and address or addresses.
Now the product number and quantity are requested. Again, as in INV1, to terminate the product input, enter a zero for the product number. Discount, product total, tax, shipping, and grand total are calculated and printed.
An example of the input subroutine is given below:

```
ENTER TAX RATE (%)? 6
ENTER DATE? 09/20/81
```

```
ENTER FIRST INVOICE NO.? 101
ENTER ACCOUNT NUMBER? 2345
DISCOUNT YES(1) NO(0)? 0
SHIPPING AUTOMATICALLY ADDED? YES(1) NO(0)?
0
HOW MUCH? 3.50
CUSTOMERS ORDER NO.? 123
NAME OF CUSTOMER? JOHN BELL INC.
NUMBER AND STREET? 991 FM STREET
CITY,STATE,ZIP-CODE?  SAN   BERNARDINO   CA
93568

1=NET 30 DAYS, 2=COD, 3=PREPAID

TERMS? 1

1=FORWARDER, 2=PARCEL SEVICE, 3=PARCEL POST

SHIPPED VIA? 2
```

If you wish your company name and address printed on the top left-hand side of the form or paper, enter them on lines 210-214 in the program.
For shipping and handling, refer to lines 390-440. Quantities smaller than 15, the charge is $1.00.
On quantities greater than 14, and smaller than 30, the charge is $1.50. If these amounts are to be changed, change the variable NV.
A similar process takes place with the calculations for the discount, which can be found on lines 360-380.
The discount is based on the quantities:

```
  1  -  5  25%
  6  - 10  33%
 11  - UP  40%
```

If it is necessary to change this information, change the variable D to the desired value. If it is to be eliminated, use another input statement which sets D to zero.



The product information and price should be entered beginning with line 865. A sample could look like this:

```
865 DATA 100, "BATTERIES 9V", 150
```

The 100 being the product number, "BATTERIES, 9V" being the product description, and 150 being the retail price.
The program also provides a shipping label with the customer's and the shipper's name. Your company address must be written into the program starting at line 960.
At the end of each day's session, type in zero when the program asks for another customer number. All quantities entered that day will be listed automatically and shown in a table.

## INVOICE 1

```
100 REM C BY ELCOMP 1980
110 DIM T(10,2)
120 INPUT"ENTER TAX RATE (%)";TR:TR=TR/100
130 INPUT"ENTER DATE";RD$
140 INPUT"ENTER FIRST INVOICE NO.";I1
150 INPUT"ENTER CUSTOMER NO.";C1
160 IF C1=0 THEN 1400
170 INPUT"NAME OF CUSTOMER";S1$
180 INPUT"STREET AND NO.";S2$
190 INPUT"CITY,STATE,ZIP-CODE";S3$
200 SAVE
210 PRINTTAB(15);"3873L SCHAEFER AVENUE,CHIN
O,CALIFORNIA 91710"
220 PRINT:PRINT:PRINT
230 PRINT TAB(5);C1
240 PRINT TAB(5);S1$
250 PRINT TAB(5);S2$
260 PRINT TAB(5);S3$
270 PRINTTAB(42);I1;TAB(63);RD$
280 PRINT:PRINT
290 PRINT
300 PRINT:PRINT:PRINT
```

```
 310 POKE517,0
 320 GOSUB 740
 330 GOSUB 490
 340 GOTO 310
 350 SAVE:S1=I
 360 IFS1>0ANDS1<6THEND=.25
 370 IFS1>5ANDS1<11THEND=.33
 380 IFS1>10THEND=.40
 390 REM S1=QUANTITY V=SHIPPING AND HANDLING
 400 IFS1<15THENV=1
 410 IFS1>14 ANDS1<30THENV=1.25
 420 IFS1>29ANDS1<50 THENV=1.5
 430 IFS1>49ANDS1<100THENV=2
 440 IFS1>99THENV=2.5
 450 REM NEXT LINE INCREMENTS INVOICE NUMBER
 460 I1=I1+1
 470 R=INT(D*100+.5)
 480 GOTO 560
 490 SAVE
 510 A=0:GOSUB 1090
 520 G$=D$:L1=L
 525 C=S1*D
 530 A=C:GOSUB 1090
 540 PRINTTAB(5);N1;TAB(12);S1;TAB(24);E$;TAB
(63-L1);G$;TAB(72-L);D$
 548 C=S1*D:T=T+C
 550 RETURN
 560 D1=D*T
 570 D1=INT(D1*100+.5)/100
 580 PRINT
 590 A=D1:GOSUB 1090
 600 PRINTTAB(46);R;TAB(50);"%DISCOUNT= -";TA
B(72-L);D$
 610 FORP=1TO(24-ZS):PRINT:NEXT
 620 A=T-D1:GOSUB 1090
 630 G$=D$:L1=L
 640 F=T-D1+V:M1=F*TR
 650 A=M1:GOSUB 1090
 660 PRINTTAB(57-L);D$;TAB(59);TR*100;"%";TAB
(72-L1);G$
 680 PRINT:PRINT:PRINT
 690 A=V:GOSUB 1090
 700 PRINTTAB(72-L);D$
```



```
710 A=M1+F:GOSUB 1090
720 PRINTTAB(72-L);D$
730 GOSUB 950:POKE517,0:GOTO 150
740 INPUT"WHICH ITEM NO.";N1
750 IF N1=0 THEN 350
760 RESTORE
770 READ N,E$,D
780 IFN=1THEN 810
790 IF N=N1 THEN 830
800 GOTO 770
810 PRINT"ITEM NO. NOT FOUND"
820 GOTO 740
830 INPUT"QUANTITY OF THIS ITEM";S1
835 GOSUB 1300
840 I=I+S1:ZS=ZS+1
850 RETURN
860 REM ITEM LIST
865 DATA 150,"CARE AND F. OF THE PET",11.00
868 DATA 151,"8K MICROSOFT MANUAL",9.95
870 DATA 152,"EXPANSION FOR 6502",9.95
875 DATA 153,"APPLICATION NOTES",14.90
880 DATA 154,"COMPLEX SOUND",6.95
885 DATA 155,"FIRST BOOK OF TRS-80",14.90
888 DATA 156,"SMALL BUSINESS PROGRAMS",14.90
890 DATA 2001,"MONJANA CBM MONITOR",98.00
895 DATA 3475,"ASSEMBLER FOR CBM",49.00
900 DATA 3476,"EDITOR/ASSEMBLER",69.00
910 DATA 3999,"BASIC F. TRS-80 (I/O)",34.50
920 DATA 1,"1",0
930 RETURN
940 END
950 PRINT:PRINT
960 PRINT"E L C O M P"
970 PRINT:PRINT"3873L SCHAEFER AVENUE"
980 PRINT:PRINT"CHINO,CALIFORNIA 91710"
990 FORZV=1TO9
1000 PRINT
1010 NEXTZV
1020 PRINTTAB(30);S1$
1030 PRINT:PRINTTAB(30);S2$
1040 PRINT:PRINT:PRINT
1050 PRINT S3$;TAB(30);S3$
1060 PRINT:PRINT:PRINT:PRINT:PRINT
```

```
1070 ZS=0
1080 RETURN
1090 K1$="0"
1100 K2$=".00"
1120 B=INT(A*100+.5)/100
1130 D1$=STR$(B)
1140 D0=B*10
1150 D2=INT(D0)
1160 D3=D0-D2
1170 IF D3<0.09THEN 1200
1180 D$=D1$
1190 GOTO 1260
1200 D4=INT(B)
1210 D5=B-D4
1220 IF D5<0.09 THEN 1250
1230 D$=D1$+K1$
1240 GOTO 1260
1250 D$=D1$+K2$
1260 L=LEN(D$)
1270 RETURN
1300 FORR1=1TO11
1310 IFT(R1,1)=N1ORT(R1,1)=0THEN1350
1320 NEXTR1
1350 T(R1,1)=N1
1360 T(R1,2)=T(R1,2)+S1
1370 RETURN
1400 PRINT:PRINT:PRINT:SAVE
1410 PRINT"ITEM NO.";TAB(12);"#SOLD"
1415 PRINT
1420 FORR1=1TO11
1425 IFT(R1,1)=0THEN1450
1430 PRINTTAB(2);T(R1,1);TAB(14);TR1,2)
1440 NEXTR1
1450 POKE517,0
1460 END
```



INVOICE 2

```
100 REM C BY ELCOMP 1980
110 DIM T(10,2)
120 INPUT"ENTER TAX RATE (%)";TR:TR=TR/100
130 INPUT"ENTER DATE";RD$
140 INPUT"ENTER FIRST INVOICE NO.";I1
150 INPUT"ENTER CUSTOMER NO.";C1
160 IF C1=0 THEN 1400
170 INPUT"NAME OF CUSTOMER";S1$
180 INPUT"STREET AND NO.";S2$
190 INPUT"CITY,STATE,ZIP-CODE";S3$
200 POKE8994,3
205 PRINTTAB(15);"PERSOCOMP,INC"
210 PRINTTAB(15);"6502 MICRO BOULEVARD,COMPO
NA,CALIFORNIA 91766"
215 PRINTTAB(15);"ORIGINAL I N V O I C E"
220 PRINT:PRINT:PRINT
230 PRINT TAB(5);C1
240 PRINT TAB(5);S1$
250 PRINT TAB(5);S2$
260 PRINT TAB(5);S3$
270 PRINTTAB(42);I1;TAB(63);RD$
280 PRINT:PRINT
290 PRINT
300 PRINT:PRINT
310 POKE8994,2
320 GOSUB 740
330 GOSUB 490
340 GOTO 310
350 POKE8994,3:S1=I
360 IFS1>0ANDS1<6THEND=.25
370 IFS1>5ANDS1<11THEND=.33
380 IFS1>10THEND=.40
390 REM S1=QUANTITY V=SHIPPING AND HANDLING
400 IFS1<15THENV=1
410 IFS1>14 ANDS1<30THENV=1.25
420 IFS1>29ANDS1<50 THENV=1.5
430 IFS1>49ANDS1<100THENV=2
440 IFS1>99THENV=2.5
450 REM NEXT LINE INCREMENTS INVOICE NUMBER
460 I1=I1+1
470 R=INT(D*100+.5)
```

```
 480 GOTO 560
 490 POKE8994,3
 510 A=0:GOSUB 1090
 520 G$=D$:L1=L
 525 C=S1*O
 530 A=C:GOSUB 1090
 540 PRINTTAB(5);N1;TAB(12);S1;TAB(24);E$;TAB
(63-L1);G$;TAB(72-L);D$
 548 C=S1*O:T=T+C
 550 RETURN
 560 D1=D*T
 570 D1=INT(D1*100+.5)/100
 580 PRINT
 590 A=D1:GOSUB 1090
 600 PRINTTAB(46);R;TAB(50);"%DISCOUNT= -";TA
B(72-L);D$
 610 FORP=1TO(24-ZS):PRINT:NEXT
 620 A=T-D1:GOSUB 1090
 630 G$=D$:L1=L
 640 F=T-D1+V:M1=F*TR
 650 A=M1:GOSUB 1090
 660 PRINTTAB(57-L);D$;TAB(59);TR*100;"%";TAB
(72-L1);G$
 680 PRINT:PRINT:PRINT
 690 A=V:GOSUB 1090
 700 PRINTTAB(72-L);D$
 710 A=M1+F:GOSUB 1090
 720 PRINTTAB(72-L);D$
 730 GOSUB 950:POKE8994,2:GOTO 150
 740 INPUT"WHICH ITEM NO.";N1
 750 IF N1=0 THEN 350
 760 RESTORE
 770 READ N,E$,O
 780 IFN=1THEN 810
 790 IF N=N1 THEN 830
 800 GOTO 770
 810 PRINT"ITEM NO. NOT FOUND"
 820 GOTO 740
 830 INPUT"QUANTITY OF THIS ITEM";S1
 835 GOSUB 1300
 840 I=I+S1:ZS=ZS+1
 850 RETURN
 860 REM ITEM LIST
```



```
865 DATA 150,"CARE AND F. OF THE PET",11.00
868 DATA 151,"8K MICROSOFT MANUAL",9.95
870 DATA 152,"EXPANSION FOR 6502",9.95
875 DATA 153,"APPLICATION NOTES",14.90
880 DATA 154,"COMPLEX SOUND",6.95
885 DATA 155,"FIRST BOOK OF TRS-80",14.90
888 DATA 156,"SMALL BUSINESS PROGRAMS",14.90
890 DATA 2001,"MONJANA CBM MONITOR",98.00
895 DATA 3475,"ASSEMBLER FOR CBM",49.00
900 DATA 3476,"EDITOR/ASSEMBLER",69.00
910 DATA 3999,"BASIC F. TRS-80 (I/O)",34.50
911 DATA 160,"VIP-FOURTH BOOK OF OHIO",9.95
920 DATA 1,"1",0
930 RETURN
940 END
950 PRINT:PRINT
960 PRINT"PERSOCOMP,INC"
970 PRINT"6502 MICRO BOULEVARD"
980 PRINT:PRINT"COMPONA,CALIFORNIA 91766"
990 FOR ZV=1TO 4
1000 PRINT
1010 NEXTZV
1020 PRINTTAB(30);S1$
1030 PRINT:PRINTTAB(30);S2$
1040 PRINT
1050 PRINTTAB(30);S3$
1060 PRINT:PRINT:PRINT:PRINT:PRINT
1070 ZS=0
1075 POKE8994,2
1080 RETURN
1090 K1$="O"
1100 K2$=".00"
1120 B=INT(A*100+.5)/100
1130 D1$=STR$(B)
1140 DO=B*10
1150 D2=INT(DO)
1160 D3=DO-D2
1170 IF D3<0.09THEN 1200
1180 D$=D1$
1190 GOTO 1260
1200 D4=INT(B)
1210 D5=B-D4
1220 IF D5<0.09 THEN 1250
```

```
1230 D$=D1$+K1$
1240 GOTO 1260
1250 D$=D1$+K2$
1260 L=LEN(D$)
1270 RETURN
1300 FORR1=1TO11
1310 IFT(R1,1)=N1ORT(R1,1)=0THEN1350
1320 NEXTR1
1350 T(R1,1)=N1
1360 T(R1,2)=T(R1,2)+S1
1370 RETURN
1400 PRINT:PRINT:POKE8994,3
1410 PRINT"ITEM NO.";TAB(12);"#SOLD"
1415 PRINT
1420 FORR1=1TO11
1425 IFT(R1,1)=0THEN1450
1430 PRINTTAB(2);T(R1,1);TAB(14);T(R1,2)
1440 NEXTR1
1450 POKE8994,2
1460 END
```

INVOICE 3

```
100 REM C BY ELCOMP 1980
105 FORQ=1TO30:PRINT:NEXT
110 DIM T(10,2)
120 INPUT"ENTER TAX RATE (%)";TR:TR=TR/100
130 INPUT"ENTER DATE";RD$
140 INPUT"ENTER FIRST INVOICE NO.";I1
150 INPUT"ENTER ACCOUNT NUMBER";C1
160 IF C1=0 THEN 1400
161 SC=1:DC=1:D=0:V=0
162 INPUT"DISCOUNT YES(1) NO(0)";DC
163 INPUT"SHIPPING AUTOMATICLY ADDED? YES(1)
NO(0)";SC
164 IF SC=0 THEN INPUT"HOE MUCH";V
165 INPUT"CUSTOMERS ORDER NO.";CO
170 INPUT"NAME OF CUSTOMER";S1$
172 INPUT"NUMBER AND STREET";S2$
174 INPUT"CITY,STATE,ZIP-CODE";S3$
176 PRINT:PRINT:PRINT
177 PRINT"1=NET 30 DAYS,2=COD,3=PREPAID"
178 PRINT:INPUT"TERMS";TE
```



```
179 IFTE=1THENTE$="NET 30 DAYS"
180 IFTE=2THENTE$="C O D"
181 IFTE=3THENTE$="PREPAID"
185 PRINT:PRINT:PRINT
186 PRINT"1=FORWARDER,2=PARCEL SERVICE,3=PAR
CEL POST"
187 PRINT:INPUT"SHIPPED VIA";SV
188 IFSV=1THENSV$="FORWARDER"
189 IFSV=2THENSV$="PARCEL SERVICE"
190 IFSV=3THENSV$="PARCEL POST"
192 FORQ=1TO30:PRINT:NEXT
194 PRINTS1$
195 PRINTS2$
196 PRINTS3$
197 PRINT:PRINT:INPUT"SHIP TO THE SAME ADDRE
SS";Q$
198 IFQ$="Y"THEN205
199 PRINT:PRINT:PRINT
200 INPUT"NAME";H1$
201 INPUT"NUMBER AND STREET";H2$
202 INPUT"CITY,STATE,ZIP-CODE";H3$
203 GOTO208
205 H1$=S1$:H2$=S2$:H3$=S3$
208 FORQ=1TO9:PRINT#1:NEXT
210 PRINT#1
212 PRINT#1,TAB(62);RD$
214 PRINT#1
220 PRINT#1:PRINT#1
230 PRINT#1,TAB(10);C1;TAB(65);I1
235 PRINT#1:PRINT#1
240 PRINT#1,TAB(8);S1$;TAB(43);H1$
250 PRINT#1,TAB(8);S2$;TAB(43);H2$
260 PRINT#1,TAB(8);S3$;TAB(43);H3$
280 PRINT#1:PRINT#1:PRINT#1
290 PRINT#1:PRINT#1
295 PRINT#1,TAB(14);CO;TAB(23)"XX"TAB(31);TE
$;TAB(47);SV$;TAB(66);RD$
300 PRINT#1:PRINT#1:PRINT#1
320 GOSUB 740
330 GOSUB510
340 GOTO320
350 S1=I
355 IF DC=0 THEN GOTO 390
```

```
 360 IFS1>0ANDS1<6THEND=.25
 370 IFS1>5ANDS1<11THEND=.33
 380 IFS1>10THEND=.40
 390 REM S1=QUANTITY V=SHIPPING AND HANDLING
 395 IF SC=0 THEN GOTO 450
 400 IFS1<15THENV=1
 410 IFS1>14 ANDS1<30THENV=1.25
 420 IFS1>29ANDS1<50 THENV=1.5
 430 IFS1>49ANDS1<100THENV=2
 440 IFS1>99THENV=2.5
 450 REM NEXT LINE INCREMENTS INVOICE NUMBER
 460 I1=I1+1
 470 R=INT(D*100+.5)
 480 GOTO 560
 510 A=0:GOSUB 1090
 520 G$=D$:L1=L
 525 C=S1*0
 530 A=C:GOSUB 1090
 538 PRINT#1,TAB(3);Z1;
 540 PRINT#1,TAB(9);S1;TAB(14);N1;TAB(20);E$;
TAB(59-L1);G$;TAB(76-L);D$
 548 C=S1*0:T=T+C
 550 RETURN
 560 D1=D*T
 570 D1=INT(D1*100+.5)/100
 580 PRINT#1
 590 A=D1:GOSUB 1090
 600 PRINT#1,TAB(44);R;TAB(48);"%DISCOUNT= -"
;TAB(76-L);D$
 610 FORP=1TO22-ZS:PRINT#1:NEXT
 620 A=T-D1:GOSUB 1090
 630 G$=D$:L1=L
 632 A=V:GOSUB1090
 634 H$=D$
 635 M1=(T-D1+V)*TR
 636 A=T-D1+M1+V:GOSUB1090
 638 J$=D$:L2=L
 650 A=M1:GOSUB 1090
 660 PRINT#1,TAB(9-L1);G$;TAB(17-L);D$;TAB(21
);H$;TAB(76-L2);J$
 680 PRINT#1:PRINT#1:PRINT#1
 730 GOSUB950:GOTO150
 740 INPUT"WHICH ITEM NO.";N1
```



```
 750 IF N1=0 THEN 350
 760 RESTORE
 770 READ N,E$,O
 780 IFN=1THEN 810
 790 IF N=N1 THEN 830
 800 GOTO 770
 810 PRINT"ITEM NO. NOT FOUND"
 820 GOTO 740
 830 INPUT"QUANTITY ORDERED";Z1
 833 INPUT"QUANTITY SHIPPED";S1
 835 GOSUB 1300
 840 I=I+S1:ZS=ZS+1
 850 RETURN
 860 REM ITEM LIST
 865 DATA 150,"CARE AND F. OF THE PET",11.00
 868 DATA 151,"8K MICROSOFT MANUAL",9.95
 870 DATA 152,"EXPANSION FOR 6502",9.95
 875 DATA 153,"APPLICATION NOTES",14.90
 880 DATA 154,"COMPLEX SOUND",6.95
 885 DATA 155,"FIRST BOOK OF TRS-80",14.90
 888 DATA 156,"SMALL BUSINESS PROGRAMS",14.90
 895 DATA 3475,"ASSEMBLER FOR CBM",49.00
 900 DATA 3476,"EDITOR/ASSEMBLER",69.00
 910 DATA 3999,"BASIC F. TRS-80 (I/O)",34.50
 911 DATA 160,"VIP-FOURTH BOOK OF OHIO",9.95
 912 DATA 159,"THE THIRD BOOK OF OHIO",7.95
 913 DATA 8097,"BLANK CASSETES SPECIAL",0.499
 914 DATA 2001,"MONJANA/1 MONOTOR",69.00
 915 DATA 157,"THE FIRST BOOK OF OHIO",7.95
 916 DATA 158,"THE SECOND BOOK OF OHIO",7.95
 917 DATA 1105,"OHIO DISK EXPANSION",20.00
 920 DATA 1,"1",0
 930 RETURN
 940 END
 950 PRINT#1:PRINT#1:PRINT#1
 955 PRINT#1
 960 PRINT#1
 970 PRINT#1
 980 PRINT#1
 990 FORZV=1TO8
1000 PRINT#1
1010 NEXTZV
1020 PRINT#1,TAB(5);H1$
```

```
1030 PRINT#1,TAB(5);H2$
1050 PRINT#1,TAB(5);H3$
1060 PRINT:PRINT:PRINT:PRINT:PRINT
1070 ZS=0:S1=0:T=0:I=0
1080 RETURN
1090 K1$="0"
1100 K2$=".00"
1120 B=INT(A*100+.5)/100
1130 D1$=STR$(B)
1140 D0=B*10
1150 D2=INT(D0)
1160 D3=D0-D2
1170 IF D3<0.09THEN 1200
1180 D$=D1$
1190 GOTO 1260
1200 D4=INT(B)
1210 D5=B-D4
1220 IF D5<0.09 THEN 1250
1230 D$=D1$+K1$
1240 GOTO 1260
1250 D$=D1$+K2$
1260 L=LEN(D$)
1270 RETURN
1300 FORR1=1TO11
1310 IFT(R1,1)=N1ORT(R1,1)=0THEN1350
1320 NEXTR1
1350 T(R1,1)=N1
1360 T(R1,2)=T(R1,2)+S1
1370 RETURN
1400 PRINT#1:PRINT#1:PRINT#1
1410 PRINT#1,"ITEM NO.";TAB(12);"#SOLD"
1415 PRINT#1
1420 FORR1=1TO11
1425 IFT(R1,1)=0THEN1460
1430 PRINT#1,TAB(2);T(R1,1);TAB(14);T(R1,2)
1440 NEXTR1
1460 END
```



# 590 board for the CD-23/CD-74

590 BOARD FOR THE CD-23/CD-74

N.C. - NON-COMPONENT SIDE
C.S. - COMPONENT SIDE

| ACTIVITY | LOCATION | VALUE |
|---|---|---|
| CUT FOIL | N.C. IC-B6, PIN 3 TO IC-B4, PIN 8 | ------ |
| ADD JUMPER | N.C. IC-B6, PIN 3 TO IC-B3, PIN 6 | ------ |
| CUT FOIL | C.S. IC-E7, PIN 13 TO IC-C8, PIN 1 | ------ |
| ADD DIODE | C.S. CATHODE TO IC-E7, PIN 13/ANODE TO IC-C8, PIN 1 | 1N914 |
| ADD DIODE | N.C. CATHODE TO IC-C8,PIN 13/ANODE TO IC-C8,PIN 1 > | 1N914 |
| ADD RESISTOR | N.C. IC-C8, PIN 1 TO +5V | 4.7K OHM |

# Memory map of OS65U and location of various parameters

MEMORY MAP OF OS65U AND LOCATION OF VARIOUS PARAMETERS

| LOCATION | | INITIAL CONTENTS | | |
|---|---|---|---|---|
| DEC | HEX | DEC | HEX | NAME OR DESCRIPTION |
| 23 | 0017 | | | TERMINAL WIDTH |
| 24 | 0018 | | | INT(TERM. WIDTH/14)*14+1 |
| 132 | 0084 | | | MEM SIZE - INT(MEM SIZE/256)*256 |
| 133 | 0085 | | | INT(MEM SIZE/256) |
| 1390 | 056E | 64 | 40 | BASIC LINE DELETE CODE - '@' |
| 1394 | 0572 | 95 | 5F | BASIC CHAR DELETE CODE - '-' |
| 1632 | 0660 | | | ORG "NEW" |
| 1658 | 067A | | | ORG "CLEAR" |
| 1862 | 0746 | | | ORG "FOR" |
| 2056 | 0808 | | | ORG "RESTORE" |
| 2073 | 0819 | | | CONTROL 'O' - ENABLE OR DISABLE LOC. |
| 2086 | 0826 | | | ORG "STOP" |
| 2088 | 0828 | | | ORG "END" |
| 2129 | 0851 | | | ORG "CONT" |
| 2155 | 086B | | | ORG "NULL" |
| 2183 | 0887 | | | ORG "GOSUB" |
| 2212 | 08A4 | | | ORG "GOTO" |
| 2257 | 08D1 | | | ORG "RETURN" |
| 2295 | 08F7 | | | ORG "DATA" |
| 2343 | 0927 | | | ORG "IF" |
| 2362 | 093A | | | ORG "REM" |
| 2378 | 094A | | | ORG "ON X GOTO" |
| 2468 | 09A4 | | | ORG "LET" |
| 2611 | 0A33 | | | ORG "PRINT" |
| 2676 | 0A74 | 13 | 0D | LOC OF 'CR' INPUT ECHO (POKE 0-INHIB.) |
| 2683 | 0A7B | 10 | 0A | LOC OF 'LF' INPUT ECHO (POKE 0-INHIB.) |
| 2797 | 0AED | | | ASCII ECHO TO INPUT STATEMENT - '?' |
| 2858 | 0B2A | | | ORG "INPUT" |
| 2888 | 0B48 | 27 | | POKE 0 TO ALLOW 'CR' ON INPUT |
| 2902 | 0B56 | | | ORG "READ" |
| 2972 | 0B9C | 58 | | POKE 13 TO ALLOW ',' ON INPUT |
| 2976 | 0BA0 | 44 | | POKE 13 TO ALLOW ':' ON INPUT |
| 3145 | 0C49 | | | ORG "NEXT" |
| 3561 | 0DE9 | | | ORG "COMPLIMENT NUMBER" |

```
|  3720 | 0E88 |      |      | ORG "LOGICAL OR"
|  3723 | 0E8B |      |      | ORG "LOGICAL AND"
|  3768 | 0EB8 |      |      | ORG "RELATIONAL OPERATOR"
|  3874 | 0F23 |      |      | ORG "DIM"
|  4304 | 10D0 |      |      | ORG "USR"
|  4612 | 1204 |      |      | ORG "FRE"
|  4645 | 1225 |      |      | ORG "POS"
|  4659 | 1233 |      |      | ORG "DEF"
|  4841 | 12E9 |      |      | ORG "STR$"
|  5478 | 1566 |      |      | ORG "CHR$"
|  5498 | 157A |      |      | ORG "LEFT$"
|  5542 | 15A6 |      |      | ORG "RIGHT$"
|  5553 | 15B1 |      |      | ORG "MID$"
|  5622 | 15F6 |      |      | ORG "LEN"
|  5637 | 1605 |      |      | ORG "ASC"
|  5671 | 1627 |      |      | ORG "VAL"
|  5768 | 1688 |      |      | ORG "PEEK"
|  5777 | 1691 |      |      | ORG "POKE"
|  5786 | 169A |      |      | ORG "WAIT"
|  5816 | 16B8 |      |      | FLOATING POINT ADD & SUB   SIZE=461
|  5825 | 16C1 |      |      | ORG "SUBTRACTION"
|  5864 | 16D8 |      |      | ORG "ADDITION"
|  6277 | 1885 |      |      | NATURAL LOG                SIZE=108
|  6323 | 18B3 |      |      | ORG "LOG"
|  6335 | 18F1 |      |      | FLOAT'G POINT MULT & DIV   SIZE=565
|  6387 | 18F3 |      |      | ORG "MULTIPLICATION"
```

```
|  6668 | 1A0C |      |      | ORG "DIVISION"
|  6950 | 1B26 |      |      | SIGN,SGN,FLOAT,NEG,ABS     SIZE=48
|  6964 | 1B34 |      |      | ORG "SGN"
|  6995 | 1B53 |      |      | ORG "ABS"
|  6998 | 1B56 |      |      | COMPARE                    SIZE=64
|  7062 | 1B96 |      |      | GREATEST INTEGER           SIZE=88
|  7111 | 1BC7 |      |      | ORG "INT"
|  7150 | 1BEE |      |      | FLOAT'G POINT INPUT        SIZE=212
|  7362 | 1CC2 |      |      | FLOAT'G POINT OUTPUT       SIZE=387
|  7749 | 1E45 |      |      | ORG "SQR" (SQR+EXP:SIZE=78)
|  7758 | 1E4E |      |      | ORG "EXPONENTIATION"
|  7815 | 1E87 |      |      | ORG "COMPLIMENT SIGN"
|  7827 | 1E93 |      |      | EXPONENTIATION             SIZE=129
|  7873 | 1EC1 |      |      | ORG "EXP"
|  7956 | 1F14 |      |      | POLYNOMIAL                 SIZE=74
|  8030 | 1F5E |      |      | RANDOM                     SIZE=68
|  8038 | 1F66 |      |      | ORG "RND"
|  8098 | 1FA2 |      |      | ORG "COS" (COS,SIN,TAN:SIZE=180)
|  8105 | 1FA9 |      |      | ORG "SIN"
|  8178 | 1FF2 |      |      | ORG "TAN"
|  8278 | 2056 |      |      | ORG "ATN"
|  8704 | 2200 |      |      | DTBLE  'END'       $0829-1=2088 D
|  8706 | 2202 |      |      | DTBLE  'FOR'       $0747-1=1362 D
|  8708 | 2204 |      |      | DTBLE  'NEXT'      $0C4A-1=3145 D
|  8710 | 2206 |      |      | DTBLE  'DATA'      $08F8-1=2295 D
|  8712 | 2208 |      |      | DTBLE  'INPUT'     $0B2B-1=2858 D
```

| | | | | | |
|---|---|---|---|---|---|
| 8714 | 220A | | DTBLE | 'DIM' | $0F23-1=3874 D |
| 8716 | 220C | | DTBLE | 'READ' | $0B57-1=2902 D |
| 8718 | 220E | | DTBLE | 'LET' | $09A5-1=2469 D |
| 8720 | 2210 | | DTBLE | 'GOTO' | $08A5-1=2212 D |
| 8722 | 2212 | | DTBLE | 'RUN' | $2423-1=9250 D |
| 8724 | 2214 | | DTBLE | 'IF' | $0928-1=2343 D |
| 8726 | 2216 | | DTBLE | 'RESTORE' | $0809-1=2056 D |
| 8728 | 2218 | | DTBLE | 'GOSUB' | $0838-1=2183 D |
| 8730 | 221A | | DTBLE | 'RETURN' | $08D2-1=2257 D |
| 8732 | 221C | | DTBLE | 'REM' | $093B-1=2362 D |
| 8734 | 221E | | DTBLE | 'STOP' | $0827-1=2086 D |
| 8736 | 2220 | | DTBLE | 'ON X GOTO' | $094B-1=2378 D |
| 8738 | 2222 | | DTBLE | 'NULL' | $086C-1=2155 D |
| 8740 | 2224 | | DTBLE | 'WAIT' | $169B-1=5786 D |
| 8742 | 2226 | | DTBLE | 'LOAD' | $28FD-1=10492 D |
| 8744 | 2228 | | DTBLE | 'SAVE' | $2997-1=10646 D |
| 8746 | 222A | | DTBLE | 'DEF' | $1234-1=4659 D |
| 8748 | 222C | | DTBLE | 'POKE' | $1692-1=5777 D |
| 8750 | 222E | | DTBLE | 'PRINT' | $0A34-1=2611 D |
| 8752 | 2230 | | DTBLE | 'CONT' | $0852-1=2129 D |
| 8754 | 2232 | | DTBLE | 'LIST' | $4BDA-1=19417 D |
| 8756 | 2234 | | DTBLE | 'CLEAR' | $067B-1=1658 D |
| 8758 | 2236 | | DTBLE | 'INDEX<' | $2C33-1=11314 D |
| 8760 | 2238 | | DTBLE | 'OPEN' | $2AD2-1=10961 D |
| 8762 | 223A | | DTBLE | 'CLOSE' | $2BC7-1=11206 D |
| 8764 | 223C | | DTBLE | 'FIND' | $4A4F-1=19022 D |

| | | | | | |
|---|---|---|---|---|---|
| 8766 | 223E | | DTBLE | 'DEV' | $2C05-1=11268 D |
| 8768 | 2240 | | DTBLE | 'FLAG' | $497F-1=18914 D |
| 8770 | 2242 | | DTBLE | 'NEW' | $0661-1=1632 D |
| 8772 | 2244 | | DTBLE | 'SGN' | $1B34 =6964 D |
| 8774 | 2246 | | DTBLE | 'INT' | $1BC7 =7111 D |
| 8776 | 2248 | | DTBLE | 'ABS' | $1B53 =6995 D |
| 8778 | 224A | | DTBLE | 'USR' | $10D0 =4304 D |
| 8780 | 224C | | DTBLE | 'FRE' | $1204 =4612 D |
| 8782 | 224E | | DTBLE | 'POS' | $1225 =4645 D |
| 8784 | 2250 | | DTBLE | 'SQR' | $1E45 =7749 D |
| 8786 | 2252 | | DTBLE | 'RND' | $1F66 =8038 D |
| 8788 | 2254 | | DTBLE | 'LOG' | $18B3 =6323 D |
| 8790 | 2256 | | DTBLE | 'EXP' | $1EC1 =7873 D |
| 8792 | 2258 | | DTBLE | 'COS' | $1FA2 =8098 D |
| 8794 | 225A | | DTBLE | 'SIN' | $1FA9 =8105 D |
| 8796 | 225C | | DTBLE | 'TAN' | $1FF2 =8178 D |
| 8798 | 225E | | DTBLE | 'ATN' | $2056 =8278 D |
| 8800 | 2260 | | DTBLE | 'PEEK' | $1688 =5768 D |
| 8802 | 2262 | | DTBLE | 'LEN' | $15F6 =5622 D |
| 8804 | 2264 | | DTBLE | 'STR$' | $12E9 =4841 D |
| 8806 | 2266 | | DTBLE | 'VAL' | $1627 =5671 D |
| 8808 | 2268 | | DTBLE | 'ASC' | $1605 =5637 D |
| 8810 | 226A | | DTBLE | 'CHR$' | $1566 =5478 D |
| 8812 | 226C | | DTBLE | 'INDEX' | $2C80 =11392 D |
| 8814 | 226E | | DTBLE | 'LEFT$' | $157A =5498 D |
| 8816 | 2270 | | DTBLE | 'RIGHT$' | $15A6 =5542 D |

```
|  8818 | 2272 |     |     | DTBLE  'MID$'      $15B1  =5553 D       |
|  8820 | 2274 | 121 | 79  | FPOT PRECEDENCE FOR ADDITION            |
|  8821 | 2275 |     |     | FPOT DISP. FOR ADDITION $16D8=5864 DEC. |
|  8823 | 2277 | 121 | 79  | FPOT PRECEDENCE FOR SUBTRACTION         |
|  8824 | 2278 |     |     | FPOT DISP. FOR SUBTRACT $16C1=5825 DEC. |
|  8826 | 227A | 123 | 7B  | FPOT PRECEDENCE FOR MULTIPLY            |
|  8827 | 227B |     |     | FPOT DISP. FOR MULTIPLY $18F3=6387 DEC. |
|  8829 | 227D | 123 | 7B  | FPOT PRECEDENCE FOR DIVISION            |
|  8330 | 227E |     |     | FPOT DISP. FOR DIVISION $1A0C=6668 DEC. |
|  8832 | 2280 | 126 | 7F  | FPOT PRECEDENCE FOR EXPONENTIATE        |
|  8833 | 2281 |     |     | FPOT DISP. FOR EXPONEN. $1E4E=7758 DEC. |
|  8835 | 2283 |  80 | 50  | FPOT PRECEDENCE FOR LOGICAL AND         |
|  8836 | 2284 |     |     | FPOT DISP. FOR LOGIC AND $0E8B=3723 D   |
|  8838 | 2286 |  70 | 46  | FPOT PRECEDENCE FOR LOGICAL OR          |
|  8839 | 2287 |     |     | FPOT DISP. FOR LOGIC OR $0E88=3720 DEC. |
|  8841 | 2289 | 125 | 7D  | FPOT PRECEDENCE FOR COMPLEMENT SIGN     |
|  8842 | 228A |     |     | FPOT DISP. FOR COMPL'T SGN $1E87=7815 D |
|  8844 | 228C |  90 | 58  | FPOT PRECEDENCE FOR COMPLIMENT NUMBER   |
|  8845 | 228D |     |     | FPOT DISP. FOR COMPL'T NUM $0DE9=3561 D |
|  8847 | 228F | 100 | 64  | FPOT PRECEDENCE FOR RELATIONAL OPERATOR |
|  8848 | 2290 |     |     | FPOT DISP. FOR RELATL OPER $0EB8=3768 D |
|  8850 | 2292 |     |     | END OF DISPATCH TABLE                   |
|  8960 | 2300 |     |     | RW LIST 'END'        TOKEN=$80=128 D.   |
|  8963 | 2303 |     |     | RW LIST 'FOR'        TOKEN=$81=129 D.   |
|  8966 | 2306 |     |     | RW LIST 'NEXT'       TOKEN=$82=130 D.   |
|  8970 | 230A |     |     | RW LIST 'DATA'       TOKEN=$83=131 D.   |
```

```
|  8974 | 230E |     |     | RW LIST 'INPUT'      TOKEN=$84=132 D.   |
|  8979 | 2313 |     |     | RW LIST 'DIM'        TOKEN=$85=133 D.   |
|  8982 | 2316 |     |     | RW LIST 'READ'       TOKEN=$86=134 D.   |
|  8986 | 231A |     |     | RW LIST 'LET'        TOKEN=$87=135 D.   |
|  8989 | 231D |     |     | RW LIST 'GOTO'       TOKEN=$88=136 D.   |
|  8993 | 2321 |     |     | RW LIST 'RUN'        TOKEN=$89=137 D.   |
|  8996 | 2324 |     |     | RW LIST 'IF'         TOKEN=$8A=138 D.   |
|  8998 | 2326 |     |     | RW LIST 'RESTORE'    TOKEN=$8B=139 D.   |
|  9005 | 232D |     |     | RW LIST 'GOSUB'      TOKEN=$8C=140 D.   |
|  9010 | 2332 |     |     | RW LIST 'RETURN'     TOKEN=$8D=141 D.   |
|  9016 | 2338 |     |     | RW LIST 'REM'        TOKEN=$8E=142 D.   |
|  9019 | 233B |     |     | RW LIST 'STOP'       TOKEN=$8F=143 D.   |
|  9023 | 233F |     |     | RW LIST 'ON'         TOKEN=$90=144 D.   |
|  9025 | 2341 |     |     | RW LIST 'NULL'       TOKEN=$91=145 D.   |
|  9029 | 2345 |     |     | RW LIST 'WAIT'       TOKEN=$92=146 D.   |
|  9033 | 2349 |     |     | RW LIST 'LOAD'       TOKEN=$93=147 D.   |
|  9037 | 234D |     |     | RW LIST 'SAVE'       TOKEN=$94=148 D.   |
|  9041 | 2351 |     |     | RW LIST 'DEF'        TOKEN=$95=149 D.   |
|  9044 | 2354 |     |     | RW LIST 'POKE'       TOKEN=$96=150 D.   |
|  9048 | 2353 |     |     | RW LIST 'PRINT'      TOKEN=$97=151 D.   |
|  9053 | 235D |     |     | RW LIST 'CONT'       TOKEN=$98=152 D.   |
|  9057 | 2361 |     |     | RW LIST 'LIST'       TOKEN=$99=153 D.   |
|  9061 | 2365 |     |     | RW LIST 'CLEAR'      TOKEN=$9A=154 D.   |
|  9066 | 236A |     |     | RW LIST 'INDEX<'     TOKEN=$9B=155 D.   |
|  9072 | 2370 |     |     | RW LIST 'OPEN'       TOKEN=$9C=156 D.   |
|  9076 | 2374 |     |     | RW LIST 'CLOSE'      TOKEN=$9D=157 D.   |
```

```
|  9081 | 2379 |      |      | RW LIST 'FIND'      TOKEN=$9E=153 D.   |
|  9085 | 237D |      |      | RW LIST 'DEV'       TOKEN=$9F=159 D.  |
|  9088 | 2380 |      |      | RW LIST 'FLAG'      TOKEN=$A0=160 D.  |
|  9092 | 2384 |      |      | RW LIST 'NEW'       TOKEN=$A1=161 D.  |
|  9095 | 2387 |      |      | RW LIST 'TAB'       TOKEN=$A2=162 D.  |
|  9099 | 238B |      |      | RW LIST 'TO'        TOKEN=$A3=163 D.  |
|  9101 | 238D |      |      | RW LIST 'FN'        TOKEN=$A4=164 D.  |
|  9103 | 238F |      |      | RW LIST 'SPC('      TOKEN=$A5=165 D.  |
|  9107 | 2393 |      |      | RW LIST 'THEN'      TOKEN=$A6=166 D.  |
|  9111 | 2397 |      |      | RW LIST 'NOT'       TOKEN=$A7=167 D.  |
|  9114 | 239A |      |      | RW LIST 'STEP'      TOKEN=$A8=168 D.  |
|  9118 | 239E |      |      | RW LIST '+'         TOKEN=$A9=169 D.  |
|  9119 | 239F |      |      | RW LIST '-'         TOKEN=$AA=170 D.  |
|  9120 | 23A0 |      |      | RW LIST '*'         TOKEN=$AB=171 D.  |
|  9121 | 23A1 |      |      | RW LIST '/'         TOKEN=$AC=172 D.  |
|  9122 | 23A2 |      |      | RW LIST '^'         TOKEN=$AD=173 D.  |
|  9123 | 23A3 |      |      | RW LIST 'AND'       TOKEN=$AE=174 D.  |
|  9126 | 23A6 |      |      | RW LIST 'OR'        TOKEN=$AF=175 D.  |
|  9128 | 23A8 |      |      | RW LIST '>'         TOKEN=$B0=176 D.  |
|  9129 | 23A9 |      |      | RW LIST '='         TOKEN=$B1=177 D.  |
|  9130 | 23AA |      |      | RW LIST '<'         TOKEN=$B2=178 D.  |
|  9131 | 23AB |      |      | RW LIST 'SGN'       TOKEN=$B3=179 D.  |
|  9134 | 23AE |      |      | RW LIST 'INT'       TOKEN=$B4=180 D.  |
|  9137 | 23B1 |      |      | RW LIST 'ABS'       TOKEN=$B5=181 D.  |
|  9140 | 23B4 |      |      | RW LIST 'USR'       TOKEN=$B6=182 D.  |
|  9143 | 23B7 |      |      | RW LIST 'FRE'       TOKEN=$B7=183 D.  |
```

```
|  9146 | 23BA |      |      | RW LIST 'POS'       TOKEN=$B8=184 D.  |
|  9149 | 23BD |      |      | RW LIST 'SQR'       TOKEN=$B9=185 D.  |
|  9152 | 23C0 |      |      | RW LIST 'RND'       TOKEN=$BA=186 D.  |
|  9155 | 23C3 |      |      | RW LIST 'LOG'       TOKEN=$BB=187 D.  |
|  9158 | 23C6 |      |      | RW LIST 'EXP'       TOKEN=$BC=188 D.  |
|  9161 | 23C9 |      |      | RW LIST 'COS'       TOKEN=$BD=189 D.  |
|  9164 | 23CC |      |      | RW LIST 'SIN'       TOKEN=$BE=190 D.  |
|  9167 | 23CF |      |      | RW LIST 'TAN'       TOKEN=$BF=191 D.  |
|  9170 | 23D2 |      |      | RW LIST 'ATN'       TOKEN=$C0=192 D.  |
|  9173 | 23D5 |      |      | RW LIST 'PEEK'      TOKEN=$C1=193 D.  |
|  9177 | 23D9 |      |      | RW LIST 'LEN'       TOKEN=$C2=194 D.  |
|  9180 | 23DC |      |      | RW LIST 'STR$'      TOKEN=$C3=195 D.  |
|  9184 | 23E0 |      |      | RW LIST 'VAL'       TOKEN=$C4=196 D.  |
|  9187 | 23E3 |      |      | RW LIST 'ASC'       TOKEN=$C5=197 D.  |
|  9190 | 23E6 |      |      | RW LIST 'CHR$'      TOKEN=$C6=198 D.  |
|  9194 | 23EA |      |      | RW LIST 'INDEX'     TOKEN=$C7=199 D.  |
|  9199 | 23EF |      |      | RW LIST 'LEFT$'     TOKEN=$C8=200 D.  |
|  9204 | 23F4 |      |      | RW LIST 'RIGHT$'    TOKEN=$C9=201 D.  |
|  9210 | 23FA |      |      | RW LIST 'MID$'      TOKEN=$CA=202 D.  |
|  9214 | 23FE |      |      | RW LIST '0'  MARKS END OF TABLE       |
|  9250 | 2422 |      |      | ORG "RUN"                             |
|  9473 | 2501 |      |      | BASOD= LOCATION OF DEVICE # FOR OUTPUT|
|  9478 | 2506 |      |      | ARC TAN                  SIZE=109     |
| 10226 | 27F1 |      |      | DISK ERROR NUMBER                     |
| 10229 | 27F5 |      |      | BASIC INPUT      BASIC I/O HOOK       |
| 10248 | 2808 |      |      | BASIC OUTPUT     BASIC I/O HOOK       |
```

| | | | | |
|---|---|---|---|---|
| 10492 | 28FC | | | ORG "LOAD" |
| 10646 | 2996 | | | ORG "SAVE" |
| 10961 | 2AD1 | | | ORG "OPEN" |
| 11206 | 2BC6 | | | ORG "CLOSE" |
| 11268 | 2C04 | | | ORG "DEV" |
| 11314 | 2C32 | | | ORG "INDEX<" |
| 11392 | 2C80 | | | ORG "INDEX" |
| 11535 | 2D0F | | | ADDR. OF INPDN (CONSOLE INPUT) |
| 11556 | 2D24 | | | ADDR. OF INPSDN (SPECIFIC DEV INPUT) |
| 11571 | 2D33 | | | ADDR. OF OUTD (CONSOLE OUTPUT) |
| 11577 | 2D39 | | | ADDR. OF OUTSD (SPECIFIC DEV OUTPUT) |
| 11657 | 2D89 | | | LSB MEMORY INPUT POINTER |
| 11658 | 2D8A | | | MSB MEMORY INPUT POINTER |
| 11661 | 2D8D | | | LSB MEMORY OUTPUT POINTER |
| 11662 | 2D8E | | | MSB MEMORY OUTPUT POINTER |
| 11664 | 2D90 | | | CONSOLE INPUT DEVICE NUMBERS |
| 11665 | 2D91 | | | CONSOLE OUTPUT DEVICE NUMBERS |
| 11666 | 2D92 | | | LSB MEM. I/O PNTR FOR INDIRECT FILES |
| 11667 | 2D93 | | | MSB MEM. I/O PNTR FOR INDIRECT FILES |
| 11668 | 2D94 | | | ACTIVE DEVICE FLAG (INPUT) |
| 11686 | 2DA6 | | | ACTIVE DEVICE FLAG (OUTPUT) |
| 11702 | 2DB6 | | | LSB CONSTANT MEM. SIZE (0=USE ALL MEM) |
| 11703 | 2DB7 | | | MSB CONSTANT MEM. SIZE (0=USE ALL MEM) |
| 11774 | 2DFE | | | PEEK(11774)+PEEK(11775)*256= |
| 11775 | 2DFF | | | LINE # WHERE DISC ERROR OCCURRED |
| 11802 | 2E1A | | | NUMBER FLOPPY DRIVES ON SYSTEM |

| | | | | |
|---|---|---|---|---|
| 12644 | 3164 | 2 | 02 | SEEK RETRY COUNT (0=NO RETRIES) |
| 12796 | 31FC | 6 | 06 | READ RETRY COUNT (0=NO RETRIES) |
| 13002 | 32CA | 2 | 02 | WRITE RETRY COUNT (0=NO RETRIES) |
| 13322 | 340A | | | NUMBER OF CD-74 HARD DISC ON SYSTEM |
| 14337 | 3833 | | | # LINES/PAGE FOR PRINTER ON PORT 5 |
| 14457 | 3879 | | | DESIRED # OF PRINTED LINES/PAGE (ON#5) |
| 14639 | 392F | | | CONTROL C ENABLE OR DISABLE FLAG |
| 14646 | 3936 | | | POKE DECIMAL 91 TO ENABLE INDIRECT FILE |
| 14721 | 3981 | | | POKE DECIMAL 24 TO ENABLE INDIRECT FILE |
| 14942 | 3A5E | | | CONSOLE PORT INPUT I/O DISTR. |
| 14948 | 3A64 | | | $A9=169>LEVEL I $4C= 76>LEVEL III |
| 14949 | 3A65 | | | $C8=200>LEVEL I $03= 3>LEVEL III |
| 14950 | 3A66 | | | $8D=141>LEVEL I $D8=216>LEVEL III |
| 14989 | 3A8D | | | CONSOLE PORT OUTPUT I/O DISTR. |
| 15908 | 3E24 | | | CURRENT # OF PRINTED LINES (PORT 5) |
| 15297 | 3BC1 | | | 440/540 OUTPUT I/O DISTR. |
| 15448 | 3C58 | | | ASCII/POLLED KEYBD. INPUT I/O DISTR. |
| 15577 | 3CD9 | | | 430 BOARD INPUT I/O DISTR. |
| 15717 | 3D65 | | | 430 BOARD OUTPUT I/O DISTR. |
| 15751 | 3D87 | | | MEMORY INPUT I/O DISTR. |
| 15792 | 3DB0 | | | MEMORY OUTPUT I/O DISTR. |
| 15868 | 3DFC | | | LINE PRINTER #5 OUTPUT I/O DISTR. |
| 16086 | 3ED6 | 19 | 13 | LOCATION OF CONTROL S |
| 16096 | 3EE0 | 17 | 11 | LOCATION OF CONTROL Q |
| 16127 | 3EFF | 03 | 03 | LOCATION OF CONTROL C |
| 16131 | 3F03 | 15 | 0F | LOCATION OF CONTROL O |

| | | | | |
|---|---|---|---|---|
| 16137 | 3F09 | 07 | 07 | LOCATION OF CONTROL D |
| 16139 | 3F0B | 23 | 17 | LOCATION OF CONTROL W |
| 18915 | 497E | | | ORG "FLAG" |
| 19022 | 4A4E | | | ORG "FIND" |
| 19418 | 4BD9 | | | ORG "LIST" |
| 19798 | 4D56 | | | CA-10X DEV INDEX (0,2,4..30)=(1,2..16) |
| 19803 | 4D5B | | | CA-10X OUTPUT |
| 19862 | 4D96 | | | CA-10X INPUT |
| 24569 | 5FF9 | | | DAY STORAGE LOCATION |
| 24570 | 5FFA | | | MONTH STORAGE LOCATION |
| 24571 | 5FFB | | | YEAR STORAGE LOCATION |
| 24576 | 6000 | | | START OF BASIC WORK SPACE |
| 48658 | BE12 | | | OS65U (48K) WARM START |
| 61438 | EFFE | | | HARD DISC CYLINDER ADDR. |
| 61439 | EFFF | | | HARD DISC CYLINDER ADDR. |

# Conditional control C

CONDITIONAL CONTROL C

```
1 REM PROGRAM NAME CONDITIONAL CONTROL C
2 REM FROM COMPUTER SHOP, SAN ANTONIO, TEXAS
10 REM PROGRAM TO DEMO HOW A LONG PRINTOUT OR OTHER TASK CAN BE
20 REM INTERRUPTED FROM THE CONSOLE WITHOUT HAVING TO PRESS RESET
30 POKE 2073,96:REM DISABLE CTRL-C
40 FOR I=1 TO 1000:REM DUMMY PROGRAM WHICH WE WILL INTERRUPT AS A DEMO
50 PRINT I,I*I,SQR(I):REM PRESS CTRL-C WHILE IN THIS LOOP
60 IF PEEK(15006) THEN 90
65 REM IF CTRL-C HAS BEEN PRESSED, THEN WE WANT TO INTERRUPT THE LOOP
70 NEXT I
80 GOTO 110
90 POKE 15006,0:REM RESET CTRL-C FLAG
100 INPUT"ABORT";A$:IF A$<>"Y" THEN 70:REM PUT IN YOUR OWN DESIRED
105 REM ACTION HERE
110 POKE 2073,76:REM ENABLE CTRL-C
120 END
```

# ELCOMP

BOOKS and SOFTWARE

## For ATARI – PET – OSI – APPLE II – 6502

**ATARI BASIC – Learning by Using**
This new book is an "Action"-Book. You do more than read it. Learn the intricacy of ATARI-BASIC thorugh the short programs which are provided. The suggestions challenge you to change and write program routines. Yes, it's exciting – Many of the programs are appropriate for beginners as well as experienced computer users. (Screen Drawings, Special Sounds, Keys, Paddles + Joysticks. Specialized Screen Routines, Graphics and Sound. Peeks and Pokes and special stuff ).
**Order-No. 164** **$9.95**

**Games for the ATARI-Computer**
How to program your own games on the ATARI. Complete listings in BASIC and Machine Language of exciting games. Tricks and hints.
**Order-No. 162** **$7.95**

**ATMONA-1**
Machine Language Monitor for the ATARI 400/800.
This powerful monitor provides you with the firmware support that you need to get the most out of your powerful system. ATMONA-1 comes on a bootable cassette. No cartridges required. Disassemble, Memory Dump HEX + ASCII, (Change Memory Locations, Blocktransfer, fill memory block, save and load machine language programs, start mach. Lang. Progr. (Printer optional).
Comes with introductionary article on how to program the ATARI computer in machine language. (Available also in ROM)
**Order-No. 7022** **$19.95**

**ATMONA-2 Superstepper**
A very powerful Tracer to explore the ATARI ROM/RAM area. Stop at previously selected address. Opcode or operand (cassette).
**Order-No. 7049** **$49.95**

**The Third Book of Ohio Scientific** is now available!
Very important information for the OSI system experimenter. Interface techniques, system expansions, accessories and much more (EPROM-Burner, 6522 I/O-card with 1K RAM, Soundboard, EPROM/RAM board).
**Order-No. 159** **$7.95**

**The Fourth Book of OHIO**
VIP-Book – Very Important Programs. Many interesting programs for OSI computers. Sorting (Binary Tree). Differential Equitation, Statistics, Astrology, Gas Consumption, Games a. s. o.
**Order-No. 160** **$9.95**

**VIP Package** – Above book plus a cassette with the programs.
**Order-No. 160 A** **$19.95**

**The Fifth book of Ohio Scientific**
Many exciting programs programming hints and tricks, Textwriter, Debugger for C1P, Games, Utilities and much more (polled keyboard)
**Order-No. 161** **$7.95**

**Invoice Writing Program** for OSI-C1PMF, C4P. Disk and Cassette, 8K RAM.
**Order-No. 8234** **$29.80**

**Mailing List** for C1PMF or C4PMF 24K RAM
250 addresses incl. phone number and parameters on one 5 1/4 disk)
**Order-No. 8240** **$29.80**

**8K Microsoft BASIC Reference Manual**
Authoritative reference for the original Microsoft 4K + 8K BASIC developed for ALTAIR and later computers including OSI, PET, TRS-80 and VIC.
**Order-No. 141** **$9.95**

**Expansion Handbook for 6502 and 6802**
S-44 Card Manual describes all of the 4.5 x 6.5 44-pin S-44 cards incl. schematics. A MUST for every 6502 system user (KIM, SYM, AIM, VIC, PET, OSI)
**Order-No. 152** **$9.95**



**EDITOR/ASSEMBLER for ATARI 800, 32K RAM**
Extremely fast and powerful Editor/Assembler. (8K Source-code in about 5 seconds) Includes ATMONA-1.
**Order-No. 7098** **$49.95**

**MACRO-Assembler**
for ATARI 800, 48K RAM
Please specify your system: RAM, disc or cassette.
**Order-No. 7099** **$89.00**

**Gunfight** – For ATARI 400/800 16K RAM, needs two joysticks, animation and sound. (8K machine language).
**Order-No. 7207** **$19.95**

**EPROM BURNER** for ATARI 400/800. Bare boards only with description, schematic + software (2716, 2732).
**Order-No. 7041** **$99.00**

**Invoice Writing** for very small business with ATARI 400/800 16K RAM.
**Order-No. 7022, cass.** **$29.85**
**Order-No. 7200, disc.** **$39.99**

**Wordprocessor for ATARI 800, 48K RAM**
**Order-No. 7210** **$29.95**

**How to connect your EPSON-Printer to the ATARI 400/800.**
Construction article with printed circuit board and software. (Screenprint and variable characters per line).
**Order-No. 7210** **$19.95**

## OSI OSI OSI OSI OSI

**The First Book of Ohio Scientific**
Introduction to OSI computers. Diagrams, hardware and software information not previously available in one compact source. 192 pages.
**Order-No. 157** **$7.95**

**The Second Book of Ohio Scientific**
Very valuable information about OSI microcomputer systems. Introduction to OS-65 D and OS-65U networking. Hardware and software hints and tips. Systems specifications. Business applications.
**Order-No. 158** **$7.95**

**Microcomputer Application Notes**
Reprint of Intel's most important application notes including 2708, 8085, 8255, 6251 chips. Very necessary for the hardware buff.
**Order-No. 153** **$9.95**

**Complex Sound Generation**
New revised applications manual for the Texas Instruments SN 76477 Complex Sound Generator.
**Order-No. 154** **$6.95**

**Small Business Programs**
Complete listings for the business user. Inventory, Invoice Writing, Mailing List and much more. Introduction to Business Applications.
**Order-No. 156** **$14.90**

**Microcomputer Hardware Handbook**
Descritions, pinouts and specifications of the most popular microprocessor and support chips. A MUST for the hardware buff.
**Order-No. 29** **$14.95**

**Care and Feeding of the Commodore PET**
Eight chapters exploring PET hardware. Includes repair and interfacing information. Programming tricks and schematics.
**Order-No. 150** **$9.95**

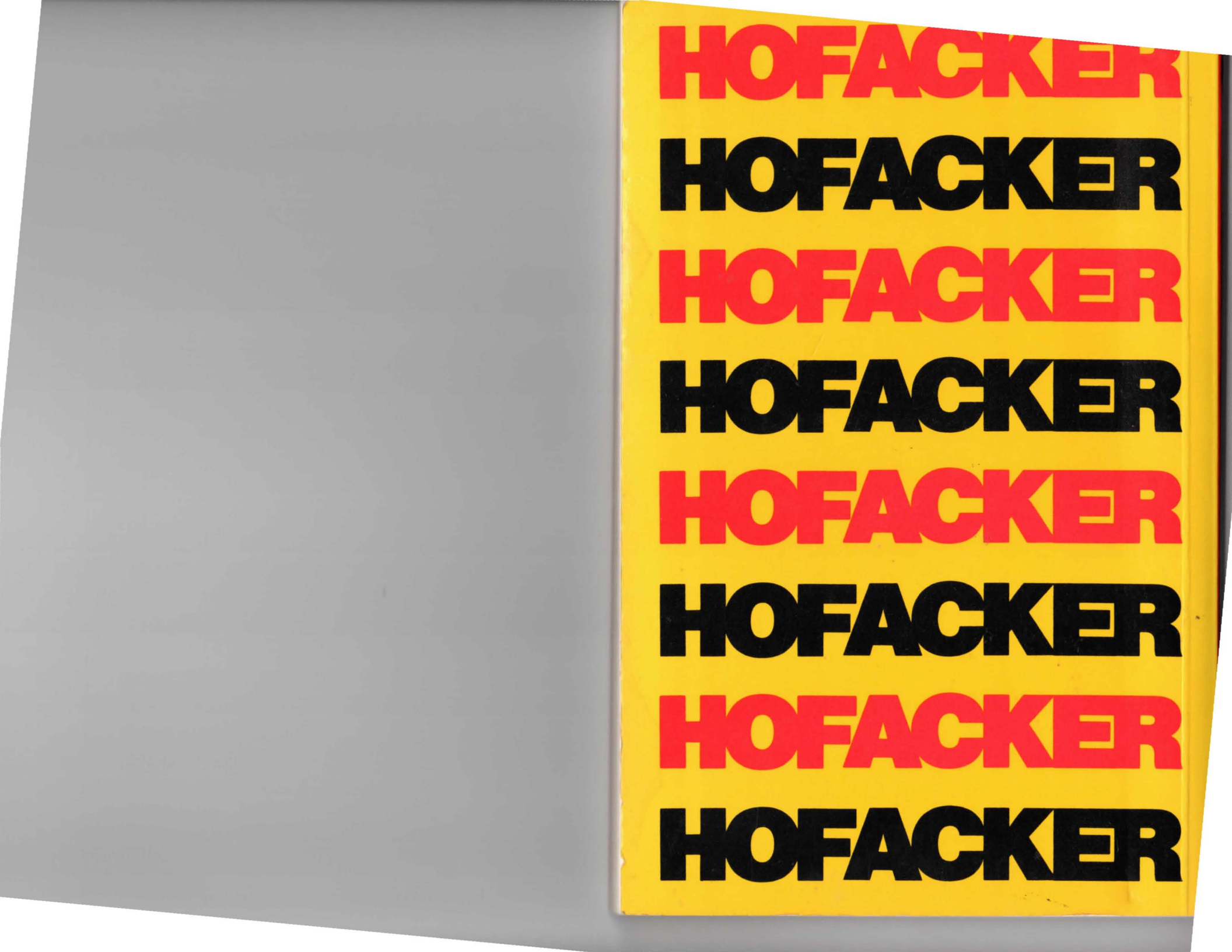
**Prototype-Expansion Board for VIC-20 (S-44-Bus).**
**Order-No. 4844** **$18.95**

**16K RAM/ROM board for S44-bus.** Any combination of RAM and ROM on one board. (SY2128 or 2716)
**Order-N0. 613** **$39.95**

**Low cost expanison boards for your APPLE II.** Bare board comes with extensive description and software.
**Prototyping card**
**Order-No. 604** **$29.00**
**6522 VIA–I/O Exp.**
**Order-No. 605** **$39.00**
**2716 EPROM-Burner**
**Order-No. 607** **$49,00**
**8K EPROM/RAM Card**
**Order-No. 609** **$29.00**

**ELCOMP Publishing, Inc.,**
**Postbox 1194 Pomona, CA 91769**

Payment: Check, Money Order, **VISA**, Mastercharge, Eurocheck. POSTPAID on PREPAID in USA. $5.00 handling fee for C.O.D. All orders outside USA: Add 15 % shipping. CA add 6.5 % sales tax.
ATARI is a registered trademark of ATARI Inc.
APPLE II is a registered trademark of APPLE Inc.


HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER