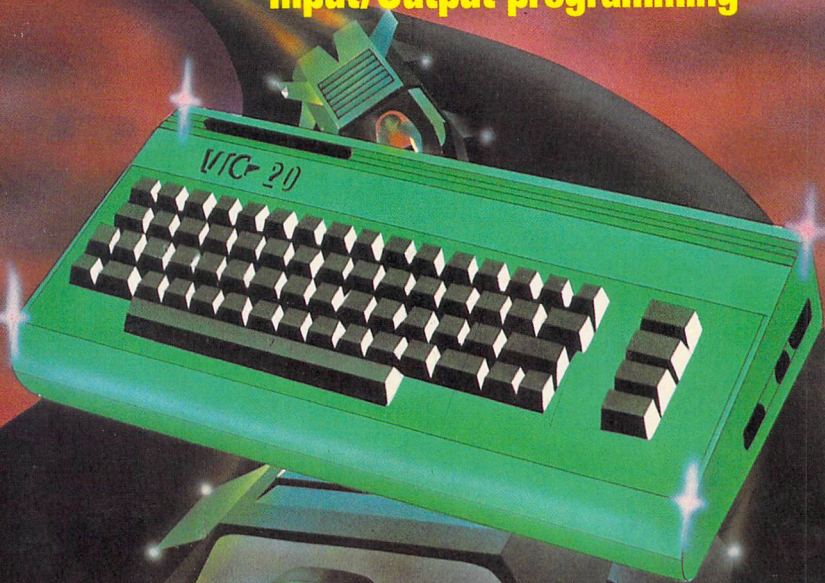


Tricks for VICs

Ready to run programs
Programming in machine language
Hardware projects
Input/Output programming



SAM D. ROBERTS

This book is an independent production of ING.W.HOFACKER GMBH International. It is published as a service to VIC-20 personal computer users worldwide.

All rights reserved. No part of this book may be reproduced by any means without the express written permission of the publisher. Example programs are for personal use only. Every reasonable effort has been made to ensure accuracy throughout this book, but neither the author or publisher can assume responsibility for any errors or omissions. No liability is assumed for any direct, or indirect damages resulting from the use of information contained herein.

First edition

First printing

January 1983 in the Federal Republic of Germany

© Copyright 1983 by Winfried Hofacker

US-Distributor:

ELCOMP PUBLISHING, INC.

Postbox 1194

Pomona, CA 91769

Phone (714) 623-8314

Publisher:

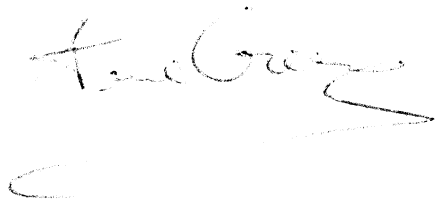
Ing. W. Hofacker GmbH

Tegernseerstr. 18, D-8150 Holzkirchen, W.-Germany

ISBN 3-88963-176-2

VIC-20 is a trademark of Commodore Business Machines, Norristown, PA

NOTE: ELCOMP PUBLISHING, INC. is not related to Commodore Business Machines.



PREFACE

The VIC-20 is a small computer. It is only small in size, but of great performance. During the last weeks, writing programs on this computer, I found out many features I liked.

The VIC-20 is a real computer, not a toy.

The programs and projects in this book range from games to mailing list and from an introduction into machine language on the VIC to input/output programming.

Though there are many ready-to-run programs in this book. Make your own changes and use the hardware hints and tricks to enhance the VIC, so it fits exactly to your specific needs.

I have to thank Franz Ende for the translation and Sefie Maier for typesetting and her help in completing this book.

Los Angeles, Spring 1983

Winfried Hofacker

TABLE OF CONTENTS

Part I: Ready to run programs

Wordprocessor for VIC-20.....	1
Test of reaction time	4
Simple number guessing game	6
Arithmetic with VIC-20	7
TIC-TAC-VIC	9
Airbattle	13
Integer divisors	16
Shooting gallery	18
Submarine	21
A clock for your VIC-20	23
Bird attack	25
Motodrom	27
Fill the aisles	30
Simple monitor program for VIC-20	32

Part II: Programming in machine-language

Programming in machine-language	38
The USR(X) function	46
Base converting with the VIC-20	49

Part III: Hardware projects

Universal experimenter board for VIC-20	54
Memory expansion for VIC-20	62
16 K byte RAMROM-board, Bytewide [®] -Concept	70

Part IV: Input/Output programming

Input/Output programming with your VIC-20	88
Input/Output programming – Part II	98
Interface for power system users	102

Appendix

How to connect up to 3 joysticks to your VIC-20	107
---	-----

Bytewide is a trademark of Mostek, Inc.

VIC-20 is a trademark of Commodore Business Machines, Inc.

Wordprocessor

for VIC-20

Wordprocessor for VIC-20 (8K RAM required)

After you have started the program use the following sequence of operations :

1. Select memory (M) or tape (T)
2. Enter line length (max. 70 characters/line)

If you selected memory (M)

3. Enter number of tabulator-positions and values for the different positions
4. When ready press the space bar.
5. Enter your text. Use key INST/DEL for corrections. The cursor keys do not work during input. If you press the key with the up-arrow the preceeding line will be displayed again.
6. When finished with your text press the key with 'fl'.
7. If you press that key again the first block of text will be displayed.
8. If you enter 'D' and a line number, that line will be displayed. Now you can edit that line. The cursor control keys can be used in this mode.
9. 'I' plus line number creates a blank line in front of that line.
10. 'D' plus line number deletes that line.
11. 'L' plus line number links that line and the preceeding line (up to max. line length)
12. '>' will display next block of text
13. '<' will display preceeding block
14. 'X' continues with program.

If you selected tape (T)

15. Enter name of file.

16. After the file has been loaded from tape the text will be displayed on the screen and you can the commands described under 8.-14.

```
90 POKE 650,128:POKE 36879,8:PRINT"█":GOSUB1560
100 PRINT"QUITEX 1█":PRINTSPC(5)"LOADING ███":GOTO400
110 PRINT"██ PRESS SPACE BAR WHEN READY"
120 GETA$:IFA$<>" "THEN120
130 RETURN
140 PRINT"██████":FORI=0TO9:IFN=I>CTHENRETURN
150 PRINT:RIGHT$(A$(N+1),LEN(A$(N+1))-1):NEXT:RETURN
160 PRINT"██████CHR$(34)CHR$(34)CHR$(20)"██SPC(A+4)"1██████":POKE631,29:POKE198,1:
RETURN
170 FORK=4096 TO4097:POKEK,32:NEXT:RETURN
180 PRINT"███████":A:"CHARACTERS/LINE":Q=1111:GOSUB320:GOSUB170:RETURN
190 PRINT"███A$""NOT POSSIBLE MEMORY FULL"
200 Q=2000:GOSUB320:GOTO900
210 GETA$:IFA$<>"Y"ANDA$<>"N"THEN210
220 RETURN
230 PRINT"██████MEMORY OR █TAP? "
240 GETB$:IFB$<>"M"ANDB$<>"T"THEN240
250 RETURN
260 FORI=1TOC:A$(I)="" :NEXT U=1:Y=FRE(0):PRINT"█":RETURN
270 PRINT"NAME OF FILE <----- IF NO NAME HIT" :INPUT"RETURN ████"A$
280 IFA$=""THENA$=""
290 PRINT"█FILE " :A$
291 IFJ=0THENPRINT"IS SEARCHED FOR":GOTO295
292 IFJ=1THENPRINT"IS SAVED":GOTO295
295 GOSUB1500:OPENB:I,J,A$:RETURN
300 B=2:I=F+1:J=1:GOSUB270:FORI=1TOC:PRINT#2,CHR$(34)+A$(I):NEXT
310 CLOSEB:POKE 193,49:RETURN
320 FORR=1TOQ:NEXT:RETURN
330 L=L+LEN(A$(N+J))*D:IFL>.7*YORC=MTHENE=2
340 RETURN
350 R=0:D=D+1:IFD>NTHEND=2:IFQ=0THENRETURN
360 IFMID$(A$,D,1)<>" " THENR=1
370 IFA$<>0THENRETURN
380 GOTO350
390 D=-1:GOSUB330:C=C-1:FORI=N+JTOC:A$(I)=A$(I+1):NEXT:A$(I)="" :RETURN
400 POKE 193,49:GOSUB230:C=B$:PRINT"█";
410 INPUT"██MAX LINE LENGTH":A:IFA=0THENA=68:IFA>70 THENPRINT"█":GOTO410
420 IFB$="T"THEN510
430 PRINT"██AUTON.CARRIAGE RET." :PRINT"BETWEEN"
440 PRINT"POSITION "A-7"AND"A+2
450 INPUT"██NUMBER OF TABS":T:IFT=0THEN500
460 IFT>10THENPRINT"█":GOTO450
470 PRINT"██DEFINE TAB POSITIONS"
480 PRINT"TAB. POSITION BETWEEN 1 AND "A:PRINT:FORI=1TOT
490 PRINT"█TAB NO." :I:INPUTT%(I):IFT%(I)>ATHEN490
491 NEXT
500 IFV=1THEN500
510 M=INT(FRE(0)/A)+10:Z=INT(.9*M):DIMA$(M):Y=FRE(0):IFB$="M"THEN500
520 B=1:I=1:J=0:GOSUB270
530 FORC=1TOZ:INPUT#1,A$(C):L=L+LEN(A$(C))
540 IFST=64THENPRINT"█FILE "A$," FOUND██":GOSUB310:GOTO870
550 IFL>.7*VTHENGOTO570
560 NEXT:C=C-1
570 F=1:POKE 193,49:GOTO860
580 GOSUB110
590 PRINT"█":POKE204,0:C=1:D=0:E=0:A$(1)=""
600 GETA$:IFA$=""THEN600
610 I=ASC(A$)AND127:IFI=13THEN750
620 IF(D=1ANDA$="" )OR(I=30RI=170RI=190RI=180RI=29)THEN600
630 D=0:IFASC(A$)=133THENPRINT"█":GOTO870
640 IFA$=","THENA$=","
650 IFI=59THENA$=","
660 IFA$=":"THENA$=":"
670 IFA$=";"THENA$=";"
680 IFI=95THENB30
690 IFI<>20ANDI<>94THENPRINTA$:GOTO790
```



```

1460 IFC#="M" THEN GOSUB 260:PRINT"O.K. CONTINUE." :Q=555:GOSUB 320:GOTO 590
1470 C#="M":GOTO 260 :V=1:GOTO 430
1480 GOSUB 260:PRINT"LOAD ATTEMPT FROM TAPE:L=0:F=0:GOSUB 110:GOTO 530
1500 PRINT:PRINT:PRINT:PRINT
1510 PRINT"PLEASE WAIT"
1520 RETURN
1550 PRINT"J":PRINT"PRINTER OUTPUT":GOSUB 1500:RETURN
1560 PRINT"J":PRINT"HEADLINE"
1565 GOSUB 210
1570 IFA#="Y" THEN 1580
1575 IFA#="N" THEN RETURN
1580 OPEN 4,4
1585 PRINT#4,CHR$(14)CHR$(17)CHR$(16)"050 INFRIED BUNTING"
1590 PRINT#4,CHR$(14)CHR$(17)CHR$(16)"45 LANGE LASSE 14"
1600 PRINT#4,CHR$(14)CHR$(17)CHR$(16)"4589 AUGSBURG"
1610 PRINT#4
1620 PRINT#4,CHR$(14)CHR$(17)CHR$(16)"44 ELEF.0821/35395"
1630 PRINT#4,CHR$(15)
1640 PRINT#4
1650 CLOSE 4:RETURN

```

Test of reaction time

3.5 K RAM

TEST OF REACTION TIME

This funny reaction time tester is the perfect game for your partys. The game is played by two players. The first player uses key A for input. The second player uses key L. If a "●" appears on the screen, the first player has to press A. The second player must not react at this time. If he does he gets a penalty. If a "○" appears on the screen, the second player has to press L. At this time, the first player must not react.

```

1 PRINT"J"
10 PRINT"TEST OF REACTION TIME"
20 PRINT"FOR TWO PLAYERS"
30 PRINT"FIRST PLAYER USES"
40 PRINT"KEY 'A',SECOND"

```

```

50 PRINT"PLAYER USES KEY 'L'"
60 PRINT"IF A '●' APPEARS"
70 PRINT"ON THE SCREEN,"
72 PRINT"THE FIRST PLAYER"
73 PRINT"HAS TO PRESS 'A'"
75 FOR V=1 TO19000:NEXT
76 PRINT"J"
80 PRINT"IF IT IS A 'O' THEN"
90 PRINT"THE SECOND PLAYER"
100 PRINT"HAS TO PRESS HIS KEY"
110 PRINT"EACH MIX-UP CAUSES 2"
120 PRINT"MINUSPOINTS"
130 PRINT"SOMETIMES A '■'"
140 PRINT"WILL BE DISPLAYED"
141 PRINT"AT THIS TIME THE FASTER"
142 PRINT"PLAYER GETS THE SCORE"
144 PRINT"IF YOU REACH SCORE"
145 PRINT"15 FIRST, YOU WIN"
150 PRINT"XPRESS ANY KEY"
160 GETA$:IFA$=""GOTO160
165 PRINT"J"
170 GOSUB200:REM DELAY
180 GOSUB300:REM PUT UP TARGET
190 GOSUB400:REM SCORE
197 GOTO170
200 REM DELAY SUBROUTINE
210 FORI=1TORND(2)*60+50:A=SIN(25):NEXT
220 RETURN
300 D=INT(RND(2)*20+1):A=INT(RND(2)*21+1)
310 PRINT"§";FORI=1TOD:PRINT"§":NEXT
320 FORI=1TOA:PRINT"■":NEXT
330 T=INT(RND(2)*3+1)
335 GETA$:IFA$<>" "THENGOTO335:REM CLR BUFFER
340 IFT=1THENPRINT"●";
350 IFT=2THENPRINT"o";
360 IFT=3THENPRINT"■";
370 RETURN
400 XM=TI
401 IFTI-60>XMTHENGOTO460
402 REM
403 REM ZEIT
405 GETA$:IFA$=""THEN GOTO401
410 IFT=1AND A$="A"THENS1=S1+1
415 IFT=1AND A$="L"THENS2=S2-2
420 IFT=2AND A$="L"THENS2=S2+1
425 IFT=2AND A$="A"THENS1=S1-2
430 IFT=3AND A$="A"THENS1=S1+1
435 IFT=3AND A$="L"THENS2=S2+1
460 FORI=1TO20:X=SIN(2):NEXT
470 GETB$:IFB$<>" "THENGOTO470:REM CLR BUFFER
480 PRINT"§§§§§PLAYER #1  PLAYER #2":PRINT" ";S1;" "§2
481 PRINT"§";FORI=1TOD:PRINT"§":NEXT
482 FORI=1TOA:PRINT"■":NEXT
483 PRINT" ";REM ERASE TARGET
490 RETURN

```

9 spaces

Simple number guessing game

SIMPLE NUMBER GUESSING GAME

In this game the computer thinks of a certain number and you have to guess the number.

After you have entered your guess, you get a hint about the relation of your and the computers number.

```
100 REM *** NUMBER GUESSING GAME ***
110 REM COPYRIGHT BY ELCOMP PUBLISHING INC.
210 PRINT"OI THINK OF A NUMBER"
215 PRINT"BETWEEN 1 AND 100"
220 PRINT"TRY TO GUESS MY"
223 PRINT"NUMBER. AFTER EACH"
225 PRINT"TRY I WILL TELL YOU"
227 PRINT"WHETHER IT WAS RIGHT,"
240 PRINT"OR TOO HIGH, OR TOO LOW"
300 REM *** COMPUTER THINKS
305 REM OF NUMBER X ***
310 LET X=INT(100*RND(1))+1
320 PRINT
330 PRINT"OK, I GOT A NUMBER"
335 PRINT"START GUESSING"
400 REM *** THE INDIVIDUAL
405 REM STARTS GUESSING ***
410 PRINT
420 PRINT"ENTER YOUR GUESS";
425 PRINT
430 INPUT G
440 IF G=X THEN 500
450 IF G>X THEN 480
460 PRINT"TOO LITTLE, TRY A"
465 PRINT"BIGGER NUMBER"
470 GOTO410
```



```

480 PRINT"TOO BIG, TRY A"
485 PRINT"SMALLER NUMBER"
490 GOTO410
500 REM ***THE INDIVIDUAL HAS FOUND
505 REM THE RIGHT NUMBER ***
510 PRINT
520 PRINT"YOU GOT IT !"
525 PRINT"AGAIN";
530 INPUT A$
535 IF A$="Y" THEN PRINT"D":GOTO 300
540 END

```

Arithmetic with

VIC-20

ARITHMETIC WITH VIC-20

With this program you can exercise either multiplication or addition. For correct answers you get a good mark. If your answer is incorrect three times, the computer will help you.

```

1 REM ARITHMETIC
10 PRINT"THIS IS AN ARITHMETIC TEST"
15 PRINT
16 PRINT "SO FAR YOU HAVE";R;"RIGHT AND";W;"WRONG"
17 PRINT
20 V=0
30 I=0
35 Z=0
40 PRINT"(1)FOR MULTIPLICATION"
50 PRINT
60 PRINT"(2)FOR ADDITION"
70 PRINT
80 INPUT I
90 PRINT
100 IF I=1 GOTO 200

```

```

110 IF I=2 GOTO 350
120 IF D=0 GOTO 500
130 GOTO 600
200 LET X=INT(RND(12)*12+1)
205 LET Y=INT(RND(12)*12+1)
210 IF X<10 THENPRINT" ";X
220 IFX>=10THENPRINT" ";X
260 IF Y<10THEN PRINT" X";Y
270 IF Y>=10THEN PRINT"X";Y
300 PRINT"_____"
310 LET Q=X*Y
320 INPUT D
330 GOTO 120
350 X=INT(RND(50)*50+1)
355 Y=INT(RND(50)*50+1)
360 IFX<10THENPRINT" ";X
370 IFX>=10THENPRINT" ";X
410 IFY<10THENPRINT"+ ";Y
420 IFY>=10THENPRINT"+ ";Y
450 PRINT"_____"
460 Q=X+Y
470 INPUTD
480 GOTO120
500 PRINT"YOU ARE RIGHT!!!":R=R+1
505 PRINT
508 Z=Z+1
509 IFZ<3GOTO512
510 GOTO10
512 IFI=1GOTO200
514 IFI=2GOTO350
600 PRINT"WRONG,TRY AGAIN...":W=W+1
610 PRINT
620 V=V+1
630 IFV=3GOTO650
640 IFI=1GOTO210
645 IFI=2GOTO360
650 PRINT"CORRECT ANSWER IS":Q
660 PRINT
670 GOTO10

```

TIC-TAC-VIC

3.5 k RAM

TIC-TAC-VIC

TIC-TAC-VIC is a VIC-20 version of TIC-TAC-TOE. When you type in the program be careful enter all the blanks and the cursor control commands!

Start the program with RUN. The computer will ask you whether you want 'X' or 'O'. The player with 'X' starts.

Try to get three of your characters in one line (vertical, horizontal, or diagonal).

```
528
3 PRINT"  ":PRINT:PRINT
4 FORX=1TO6:PRINT"      3  3      3  3":NEXTX
5 PRINT"  13  3      23  3      3"
6 PRINT"  3      3"
7 FORX=1TO6:PRINT"      3  3      3  3":NEXTX
8 PRINT"  43  3      53  3      6"
9 PRINT"  3      3"
10 FORX=1TO6:PRINT"      3  3      3  3":NEXTX
11 PRINT"  73  3      83  3      9"
50 PRINT"DO YOU WANT 'X' OR 'O'?"
52 GETC$:IFC$=""THEN52
55 IFC$="X"THEN475
60 P$="O":Q$="X"
100 G=-1:H=1:IFS(5)<>0THEN103
102 S(5)=-1:GOTO195
```

```

103  IFS(5)◇>1THEN106
104  IFS(1)◇>0THEN110
105  S(1)=-1:GOTO195
106  IFS(2)=1ANDS(1)=0THEN181
107  IFS(4)=1ANDS(1)=0THEN181
108  IFS(6)=1ANDS(9)=0THEN189
109  IFS(8)=1ANDS(9)=0THEN189
110  IFG=1THEN112
111  GOTO118
112  J=3*INT((M-1)/3)+1
113  IF3*INT((M-1)/3)+1=MTHENK=1
114  IF3*INT((M-1)/3)+2=MTHENK=2
115  IF3*INT((M-1)/3)+3=MTHENK=3
116  GOTO120
118  FORJ=1TO7STEP3:FORK=1TO3
120  IFS(J)◇>GTHEN130
122  IFS(J+2)◇>GTHEN135
126  IFS(J+1)◇>0THEN150
128  S(J+1)=-1:GOTO195
130  IFS(J)=HTHEN150
131  IFS(J+2)◇>GTHEN150
132  IFS(J+1)◇>GTHEN150
133  S(J)=-1:GOTO195
135  IFS(J+2)◇>0THEN150
136  IFS(J+1)◇>GTHEN150
138  S(J+2)=-1:GOTO195
150  IFS(K)◇>GTHEN160
152  IFS(K+6)◇>GTHEN165
156  IFS(K+3)◇>0THEN170
158  S(K+3)=-1:GOTO195
160  IFS(K)=HTHEN170
161  IFS(K+6)◇>GTHEN170
162  IFS(K+3)◇>GTHEN170
163  S(K)=-1:GOTO195
165  IFS(K+6)◇>0THEN170
166  IFS(K+3)◇>GTHEN170
168  S(K+6)=-1:GOTO195
170  GOTO450
171  IFS(3)=GANDS(7)=0THEN187
172  IFS(9)=GANDS(1)=0THEN181
173  IFS(7)=GANDS(3)=0THEN183
174  IFS(9)=0ANDS(1)=GTHEN189
175  IFG=-1THENG=1:H=-1:GOTO110

```



```

176 IFS(9)=1ANDS(3)=0THEN182
177 FORI=2TO9: IFS(I) <> 0THEN179
178 S(I)=-1:GOTO195
179 NEXTI
181 S(1)=-1:GOTO195
182 IFS(1)=1THEN177
183 S(3)=-1:GOTO195
187 S(7)=-1:GOTO195
189 S(9)=-1
195 PRINT"THE COMPUTER GOES TO.."
202 GOSUB1000
205 GOTO500
450 IFG=1THEN465
455 IFJ=7ANDK=3THEN465
460 NEXTK,J
465 IFS(5)=GTHEN171
467 GOTO175
475 P$="X":Q$="O"
500 PRINT"YOUR DRAW ?"
501 GETC$:IFC$=""THEN501
502 M=VAL(C$)
505 PRINT"!"
506 IFM=0THENPRINT"THIS WAS CUTE!"
    THANK YOU ":FORQ=1TO2000:NEXTQ:GOTO2001
507 IFM>9THEN509
508 IFS(M)=0THEN510
509 PRINT"OCCUPIED !!!"
    FORQ=1TO500:NEXTQ:GOTO500
510 G=1:S(M)=1
520 GOSUB1000
530 GOTO100
1000 GOSUB3000
1095 FORI=1TO7STEP3
1100 IFS(I) <> S(I+1)THEN1115
1105 IFS(I) <> S(I+2)THEN1115
1110 IFS(I)=-1THEN FORQ=1TO2000:NEXTQ:GOTO 1350
1115 NEXTI:FORI=1TO3: IFS(I) <> S(I+3)THEN1150
1130 IFS(I) <> S(I+6)THEN1150
1135 IFS(I)=-1THEN1350
1137 IFS(I)=1THEN1200
1150 NEXTI:FORI=1TO9: IFS(I)=0THEN1155
1152 NEXTI:GOTO1400

```

```

1155 IFS(5)<>GTHEN1170
1160 IFS(1)=GANDS(9)=GTHEN1180
1165 IFS(3)=GANDS(7)=GTHEN1180
1170 RETURN
1180 IFG=-1THEN1350
1350 FORQ=1TO2000:NEXTQ
1360 PRINT"JI WIN !":GOTO2001
1400 FORQ=1TO2000:NEXTQ
1410 PRINT"JTHIS WAS FINE, THANK YOU"
2001 INPUT"ANOTHER GAME";F$
2002 IF F$="Y" THEN RUN
2004 END
3000 FORI=1TO9
3005 PRINT"8":IFI<4THENAB=0:GOTO3040
3010 IFI>6THENAB=14:GOTO3040
3020 AB=7
3040 FORX=1TOAB+1:PRINT:NEXTX
3050 IFI=10RI=40RI=7THENBA=0:GOTO3071
3060 IFI=20RI=50RI=8THENBA=7 :GOTO3071
3070 BA=14
3071 IFP$="X"THEN3080
3072 IFS(I)=-1THENGOSUB4020
3073 IFS(I)=1THENGOSUB5020
3074 NEXTI:RETURN
3080 IFS(I)=-1THENGOSUB5020
3090 IFS(I)=1THENGOSUB4020
3095 NEXTI:RETURN
4020 PRINTTAB(BA)"  3  3  3"
4050 PRINTTAB(BA)"  3  3  3"
4060 PRINTTAB(BA)"  3  3  3"
4070 PRINTTAB(BA)"  3  3  3  3  3"
4080 PRINTTAB(BA)"  3  3  3  3  3"
4090 RETURN
5020 PRINTTAB(BA)"  3  3  3"
5050 PRINTTAB(BA)"  3  3  3  3  3"
5060 PRINTTAB(BA)"  3  3  3  3  3"
5070 PRINTTAB(BA)"  3  3  3  3  3"
5090 RETURN

```

Airbattle

8K RAM

AIRBATTLE

This game is played by two players. The left player shoots with 'A', the right player shoots with 'O'. Different planes at different heights are flying over the playfield.

Since the whole program is written in BASIC it is a little bit slow. If you change the movements and the sound generation to machine language you can develop the game.

This program shows the beginner how to create his own games.

```
1296
7 PRINT"Q";
8 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
9 FORI=1TO20:PRINT"*":NEXTI:PRINT
10 PRINT"* SHOOT OFF PLANES *"
11 FORI=1TO20:PRINT"*":NEXTI:PRINT
12 GET W$:IFW$=""THEN13
13 PRINT"Q"
14 PRINT"YOU CAN SHOOT AT 4 DIFFERENT AIMS"
15 PRINT"AT DIFFERENT TIMES."
16 PRINT"THE AIMS ARE IN DIFFERENT ALTITUDES."
17 PRINT:PRINT"THE 1ST PLAYER USES A TO SHOOT"
18 PRINT:PRINT"THE 2ND PLAYER USES O TO SHOOT"
19 PRINT:PRINT"SCORE 21 WINS":PRINT
20 PRINT:PRINT"HIT ANY KEY TO START"
21 GETZ$:IFZ$=""GOTO35
22 PRINT"Q":CLR
23 FOR I=1TO19:PRINT:NEXTI
24 PRINT"  I";TAB(17);"  I"
25 PRINTTAB(4);"——";TAB(17);"——"
26 PRINT"A=FIRE";TAB(14);"O=FIRE"
27 PRINT"Q"
28 X=INT(RND(1)*10)+3:S=0
29 X3=7702+X*22
30 FORI=1TOX:PRINT"X":NEXTI
31 X1=INT(RND(1)*4)+1
32 ON X1 GOSUB 200,300,400,460,460
```

```

125 PRINT"S"
130 GOTO100
200 FORI=1TO18
201 : IFS=1GOTO230
205 : PRINT"  71 700000";
210 : PRINT"  3 000000";:E=4
211 : IFA$="A"GOTO220
212 : IFB$="0"GOTO220
213 : T=TI
215 : IFTI-T<16-XGOTO215
220 : PRINT"  00 0";:X3=X3+1
225 : GOSUB800
230 NEXTI
240 PRINT"      000000"
250 RETURN
300 FORI=1TO18
301 : IFS=1GOTO330
305 : PRINT"  3 000000";
310 : PRINT"  7 000000";:E=4
311 : IFA$="A"GOTO320
312 : IFB$="0"GOTO320
313 : T=TI
315 : IFTI-T<15-XGOTO315
320 : PRINT"  00 0";:X3=X3+1
325 : GOSUB800
330 NEXTI
340 PRINT"      000000"
350 RETURN
400 FORI=1TO18
401 : IFS=1GOTO430
405 : PRINT"  3 000000";
410 : PRINT"  7 000000";:E=4
411 : IFA$="A"GOTO420
412 : IFB$="0"GOTO420
413 : T=TI
415 : IFTI-T<15-XGOTO415
420 : PRINT"  00 0";:X3=X3+1
425 : GOSUB800
430 NEXTI
440 PRINT"      000000"
450 RETURN
460 FORI=1TO18
461 : IFS=1GOTO490

```



```

465 : PRINT"0";
466 FOR Q=1 TO 20:NEXTQ
470 : PRINT"->|||||";E=2
471 : IFA$="A"GOTO480
472 : IFB$="0"GOTO480
473 : T=TI
475 : IFTI-T<5-XGOTO475
480 : PRINT" ||| |";X3=X3+1
485 : GOSUB800
490 NEXTI
495 PRINT"      |||||0      "
496 RETURN
600 IFZ$="A"THENA$="A"
610 IFZ$="0"THENB$="0"
620 RETURN
700 REM FIRING ROUTINE
701 P=INT(RND(1)*3)+1
702 FORD=1TOP
705 : IFA$<>"A"GOTO720
706 : IFY=0GOTO708
707 POKE38801,2:POKE8081-Y,32
708 POKE38779,2:POKE8059-Y,43:Y=Y+22
709 : IF8081-Y-X3>EGOTO715
710 : IF8081-Y-X3>=0 GOTO900
715 IFY>374 THEN A$="":POKE38779,2:POKE8081-Y,32:Y=0
718 NEXTD
720 P=INT(RND(1)*3)+1
722 FORD=1TOP
723 : IFB$<>"0"GOTO760
735 : IFZ=0GOTO750
740 POKE38814,2:POKE8094-Z,32
750 POKE38814,2:POKE8072-Z,43:Z=Z+22
752 : IF8094-Z-X3>EGOTO755
753 : IF8094-Z-X3>=0 GOTO950
755 : IFZ>374 THEN B$="":POKE38792,2:POKE8094-Z,32:Z=0
759 NEXTD
760 RETURN
800 IFA$="A"THEN GOSUB700
805 IFB$="0"THEN GOSUB700
810 GETZ$:IFZ$=""THENRETURN
820 IFZ$="A"THENA$="A":GOSUB7000:RETURN
830 IFZ$="0"THENB$="0":GOSUB7000:RETURN
840 RETURN
900 REM TARGET HIT
910 PRINT"*****|||||*****|||||";GOSUB7005
920 A$="1":AS=AS+1:Y=0:S=1:POKE59467,0
921 PRINT"      |||||";
925 PRINT"0":PRINTTAB(0);"SCORE";AS
930 IFAS=21THEN N=9:GOTO1000
940 RETURN
950 PRINT"*****|||||*****|||||";GOSUB7005
960 B$="1":BS=BS+1:Z=0:S=1:POKE59467,0
961 PRINT"      |||||";

```

```

965 PRINT"8":PRINTTAB(13);"SCORE";BS
970 IFBS=21THEN N=18:GOTO1000
975 RETURN
1000 PRINT"80";
1005 FORI=1TO10
1010 : PRINTTAB(N);"WINNER";:T9=TI
1015 : IFTI-T9<50GOTO1015
1016 : PRINTTAB(N);"*****WINNER*****";
1020 NEXTI
1025 PRINT
1030 GOTO90
7000 POKE36878,15:POKE36876,220:FORL=1TO5:NEXTL
7001 POKE36876,0:FORL=1TO500:NEXTL:POKE36878,0:RETURN
7002 POKE36878,15:POKE36876,200:FORL=1TO5:NEXTL
7003 POKE36876,0:FORL=1TO500:NEXTL:POKE36878,0
7004 RETURN
7005 POKE36877,220
7006 FORL=6TO0STEP-1:POKE36878,L:FORM=1TO12
7007 NEXTM:NEXTL:POKE36877,0
7008 POKE36878,0:RETURN
7010 IFTI-T9<30GOTO7010
7015 RETURN

```

Integer divisors

3.5 k RAM

INTEGER DIVISORS

This small program can be helpful with the homework of the student. It calculates the integer divisors of a row of numbers.

```

0 DIM F(32)
1 GOSUB 50
2 IF (10000-P)*(P-1)<0 THEN 7
3 IF P<>INT(P) THEN 7
4 IF (10000-Q)*(Q-1)<0 THEN 7
5 IF Q<>INT(Q) THEN 7
6 GOTO 8
7 GOTO 1
8 IF Q=>P THEN 12

```

```

9 LET P9=P
10 LET F=0
11 LET Q=P9
12 PRINT
13 PRINT
14 FOR J=P TO Q
15 LET F=0
16 LET M=0
17 LET R=SQR(J)
18 PRINT
19 PRINT J;TAB(8);
20 FOR K=1 TO R
21 LET X=J/K
22 IF X>INT(X) THEN 34
23 LET G=LEN(STR$(K))+2
24 IF M+G<=64 THEN 28
25 PRINT
26 PRINT TAB(8);
27 LET M=0
28 PRINT K;
29 LET M=M+G
30 IF X>R THEN 32
31 GOTO 35
32 LET F=F+1
33 LET F(F)=X
34 NEXT K
35 FOR K=F TO 1 STEP -1
36 LET G=LEN(STR$(F(K)))+2
37 IF M+G<=64 THEN 41
38 PRINT
39 PRINT TAB(8);
40 LET M=0
41 PRINT F(K);
42 LET M=M+G
43 NEXT K
44 PRINT
45 NEXT J
46 PRINT
47 PRINT
48 PRINT
49 GOTO 1
50 PRINT"THIS PROGRAM CALCULATES"
60 PRINT"THE INTEGER"

```

```

70 PRINT"DIVISORS OF TWO"
80 PRINT"VALUES"
90 PRINT"ENTER BOTH VALUES,"
92 PRINT"SEPARATED BY COMMA"
100 INPUT P,Q
101 IF P=0 THEN END
110 RETURN

```

Shooting gallery

3.5 K RAM

SHOOTING GALLERY

This is a enjoyable shooting game with sound. A firing base is moving up and down on the left side of the screen. On the right side is a wall with up to three holes. If you hit a hole you get 100 points. If you miss the hole then 10 points will be subtracted from your score. One game is five rounds. At the end of the game the total score is displayed.

```

110 REM GALLERIE      STOP
120 REM              ELCOMP PUBLISHING,INC.
130 REM              POMONA,CALIFORNIA STOP
190 REM
200 CLR
210 DIM Z(23)
220 PRINT"*****TO START HIT"
225 PRINT:PRINT"ANY KEY"
230 PRINT"PRESS 1":PRINT"FOR FIRE"
240 PRINT"*****ELCOMP MICROCOMPUTER"
250 GETO$:IF O$=""GOTO250
260 PRINT"?"
270 A$(0)=" *****ATTN"
280 A$(1)="*****ATTN*****"
290 D$(0)="X"
300 D$(2)=" "
310 G$(0)="I":G$(1)=" "
330 PRINT"S"

```

```

340 T=T+S
350 K=1:X=1:S=100:C=10:R=R+1
360 R$=STR$(R)
370 IFR=5GOTO1040
380 FORI=0TO 18:PRINT"  ":Z(I)=0:NEXT
390 FORI=1TO3
400 Z(INT(RND(1)*17+2))=1
410 NEXT
420 PRINT"■"
430 FORI=1TO19
440 PRINTTAB(19);G$(Z(I))
450 NEXT
460 PRINT"  3ROUND 3COUNT 3SCORE■"
470 PRINT"  ";R$;"      10      100  "
490 PRINT"■"
500 K=1
510 K=ABS(K-1):X=1
520 I=1
530 PRINTA$(K)
540 GET A$
550 IFA$="1"THENGOSUB640
560 IFC=0GOTO330
570 I=I+1
580 IFI=19GOTO510
590 GOTO530
600 POKE36877,220
610 FORL=6 TO0STEP-1:POKE36878,L:FOR M=1TO12
615 NEXTM:NEXTL:POKE36877,0
617 POKE36878,0
620 RETURN
640 PRINT"■■■";
650 Y=18
660 IFK=0THENW=I+2:GOTO680
670 W=20-I
680 IFZ(W)>0THENY=20
690 PRINT"■■ ●";
700 X=X+1
710 IFX<>YGOTO690
720 IFX=18THEN PRINT"■■ ■■";:GOSUB600:GOTO 840
730 Z(W)=Z(W)+1
740 FORJ=1TO4
750 PRINT"■■3■";
760 POKE59467,16

```

```

770 POKE59464,150:POKE59466,31
780 FORM=1T020:NEXT
790 PRINT"II";
820 NEXT
825 GOSUB1300
830 S=S+120
840 S=S-5*(20-X)
850 POKE59467,0
860 S=S-Z(W)*10
870 C=C-1
880 FORJ=1T022-W
890 PRINT
900 NEXT
910 IFS<0THENS=0
920 S$=STR$(S):C$=STR$(C)
930 PRINTTAB(8)"I":C$:TAB(15);RIGHT$(S$,3)
940 PRINT"X"
950 FORJ=1TOW:PRINT"X":NEXT
960 PRINT"XXXXXXXXXXXXXXXXXXXX";D$(20-X);
970 IFX<20GOTO1020
1020 X=1:PRINT"II IIT"
1030 RETURN
1040 PRINT"I"
1070 PRINT"XXX"
1080 FORJ=1T08
1090 PRINTTAB(3);"** GAME OVER BOY"
1130 PRINTTAB(3);"**GAME OVER*IT"
1170 NEXT
1180 PRINT:PRINT
1190 FORH=1T039:PRINT"♦":NEXT:PRINT
1200 POKE59467,0:POKE59466,0
1210 PRINT"XWITH 40 SHOTS ON 3 "
1215 PRINT"TARGETS"
1220 PRINT"XTOTAL SCORE"
1225 PRINT"IS";T;"POINTS"
1260 POKE59467,0:POKE59466,0
1270 PRINT"ONCE AGAIN?"
1280 INPUT A$
1290 IF A$="Y" THEN 200
1295 END
1300 REM BELL
1310 POKE 36878,15
1320 FOR L=1T05:FOR M=1T05 :POKE 36876,220

```

```

1330 FOR N=1TO5: NEXT N:POKE 37876,0:NEXT M
1340 FOR M=1 TO 10 :NEXT M:NEXT L
1350 POKE 36878,0
1360 RETURN
2000 PRINT"§"
2010 FORL=1TO11:PRINT"":NEXT

```

Submarine

3.5 K RAM

SUBMARINE

This program works with the two cursor control keys. After you RUN the program you move the cursor to the square (submarine) in the middle of the screen. If you are in the center of the submarine it will explode.

Try to get as many points as possible within 120 seconds. If you are very good you will get extra time.

```

272
50 GOSUB11000:REM PRINT INSTRUCTIONS
90 REM
99 AT=120
100 GOSUB1000:REM SETUP FIELD
105 CX=3:CY=2:PRINT"§";
110 PRINT"■■■■";
200 GOSUB2000:REM SETUP CLOCK
300 GOSUB3000:REM PUT UP TARGET
400 GOSUB4000:REM ALLOW CURSOR MOVEMENT
500 GOSUB5000:REM SCORE HIT
600 GOTO300:REM DOIT AGAIN UNTIL TIME'S UP
1000 PRINT"□";
1010 FORI=1TO22:PRINT"§ ";:NEXT
1020 FORI=1TO21:PRINT"■■§ ";:NEXT
1030 FORI=1TO21:PRINT"■■■§ ";:NEXT

```



```

5525 NEXTM:NEXTL:POKE36877,0:POKE36878,0
5530 RETURN
6000 PRINT"#####GAME OVER...WANT TO PLAY AGAIN";
6100 INPUTYN$:IFLEFT$(YN$,1)="Y"THENGOTO90
6210 END
10000 REM BONUS SECTION
10100 AT=180
10110 PRINT"#####BONUS"
10115 PRINT"@";
10120 RETURN
11000 PRINT"7":PRINT"SIMPLE SUBMARINE"
11005 PRINT"      FIGHT"
11010 PRINT"YOU WILL GET POINTS"
11020 PRINT"IF YOU BRING THE TOR-"
11022 PRINT"PEDO IN THE CENTER"
11025 PRINT"OF THE ENEMIES SUB-"
11030 PRINT"MARINE.YOU CAN USE"
11040 PRINT"THE CURSOR CONTROL"
11050 PRINT"KEYS(CSR).YOU GOT 120"
11060 PRINT"SECONDS TIME.A SCORE"
11070 PRINT"OVER 5000 GIVES YOU A"
11080 PRINT"30 POINTS BONUS."
11090 PRINT"YOU MUST HIT THE      TARGET IN THE CENTER"
11095 PRINT
11100 PRINT"HIT ANY KEY TO START...."
11110 GETA$:IF A$=""THEN GOTO 11110
11120 RETURN

```

A clock for your

VIC-20

3.5 K RAM

A CLOCK FOR YOUR VIC-20

The following program makes a clock out of your VIC-20. Hours, minutes, and seconds are displayed and a ticking is created. To set the clock you have to enter hours, minutes, and seconds in a six digit number. For example: 195352, means it is 7 hours, 53 minutes, & 52 seconds, in the p.m.

```

1 DIMA(6):DIMB(6):GOSUB3000
2 PRINT"Q";
3 POKE36879,12
4 FORI=1TO6:B(I)=-1:NEXT
10 FORI=1TO6:A(I)=VAL(MID$(TI$,I,1)):NEXTI
30 FORM=1TO6
35 PRINT"#####";
40 IFB(M)=A(M)THENGOTO100
50 PRINTTAB((M-1)*3+1):GO:UB1010:REM ERASE
55 PRINT"#####";
60 PRINTTAB((M-1)*3+1);
70 B(M)=A(M):X=A(M)
75 Z=VAL(RIGHT$(TI$,1))
78 IFZ<11 THEN GOSUB 4000
99 ON(X+1)GOSUB1009,1000,1001,1002,1003,1004,1005,1006,1007,1008,1009
100 NEXTM
200 GOTO10
1000 PRINT"#####":RETURN
1001 PRINT"#####":RETURN
1002 PRINT"#####":RETURN
1003 PRINT"#####":RETURN
1004 PRINT"#####":RETURN
1005 PRINT"#####":RETURN
1006 PRINT"#####":RETURN
1007 PRINT"#####":RETURN
1008 PRINT"#####":RETURN
1009 PRINT"#####":RETURN
1010 FORI=1TO7:PRINT"#####":NEXT:RETURN
3000 REM CLOCK SET ROUTINE
3010 PRINT"##### CLOCK FOR VIC-20"
3020 PRINT"THE TIME IS DISPLAYED"
3030 PRINT"AND ENTERED"
3035 PRINT"AS A SIX-DIGIT NUMBER"
3036 PRINT"IF THE TIME IS ALREADY"
3040 PRINT"SET, THEN ENTER RETURN";
3045 PRINT"ENTER TIME"
3050 INPUT N
3055 IFN=0THENGOTO3070
3056 IFN<100000THENTI$="0"+RIGHT$(STR$(INT(N)),5)
3060 IFN>=100000THENTI$=RIGHT$(STR$(INT(N)),6)
3070 RETURN
4000 REM CLOCK
4001 POKE36878,15
4002 REM FOR L=1TO10
4009 POKE36878,200
4030 POKE36874,0
4060 REM NEXTL
4070 POKE36878,0
4080 RETURN

```

Caution : For the next three games you need at least 8k of RAM.

Bird attack

8K RAM

BIRD ATTACK

In this game you are a water-carrier. You try to carry as many liters of water from a spring to a tub as possible, but watch out for birds dropping little bombs. The player can be controlled by key 'l' for left and key 'r' for right.

```

5 POKE36878,15
6 PRINT"3"
7 PRINT"PLEASE WAIT!"
8 REM *****
9 REM * *
10 REM* BIRD ATTACK *
11 REM* *
12 REM* *
13 REM*****
14 REM *LOAD CHARACTER GENERATOR*
15 POKE51,255:POKE52,19:POKE55,255:POKE56,19:CLR
17 PRINT"*****CONTROL WITH '<'<(LEFT) AND '>'>(RIGHT)"
20 FORI=1TO1024
30 POKE5120+I,PEEK(32768+I)
40 NEXT
50 FORI=0TO199
60 READA
70 POKE5144+I,A
80 NEXT
90 POKE36869,253:POKE36866,PEEK(36866)OR128
100 REM *BIRD*
110 DATA28,193,243,255,255,243,193,128
120 DATA255,255,255,255,255,255,255,255
130 DATA129,230,248,208,248,228,194,128
140 DATA48,56,120,124,124,126,126,126
150 DATA126,126,126,124,124,120,56,48
160 REM *WATER CARRIER*
170 DATA12,30,59,62,28,30,12,31
180 DATA62,94,158,30,10,17,33,49

```

```

190 DATA128,64,48,16,56,56,56,56
200 DATA31,30,30,30,14,12,12,14
210 DATA48,120,220,124,60,120,48,120
220 DATA252,122,121,120,80,136,132,140
230 DATA31,16,56,56,56,56,0,0
240 DATA248,120,120,120,112,48,48,112
250 REM *SPRING*
260 DATA221,221,0,247,247,0,187,187
270 REM *BULLET*
280 DATA8,8,8,8,8,8,8,8
290 REM *TUB*
300 DATA255,255,0,255,255,0,255,255
310 FORI=828T01022:READA:POKEI,A:NEXT:POKE1000,0
320 PRINT"O":FORI=38840T038883:POKEI,6:NEXT:POKE1004,0:POKE1001,18
330 POKE8160,138:POKE8159,139:POKE8138,137
350 REM *DEAD*
360 DATA0,0,0,15,143,255,255,15
370 DATA0,0,4,174,251,255,255,190
380 DATA0,0,28,28,28,28,28,0
390 REM *GROUND*
395 DATA54,255,255,255,255,255,255,255
400 REM *TREE*
405 DATA0,0,224,248,255,255,255,255
410 DATA0,0,0,0,192,192,192,240
415 DATA240,252,254,255,255,255,255,255
420 DATA255,255,255,254,248,240,240,128
425 DATA120,120,120,120,120,120,120,120
450 FORI=38510T038839:POKEI,4:NEXT:FORI=38400T038509:POKEI,7:NEXT
465 POKE8164,152:POKE8142,152:POKE8120,129:POKE8098,129:POKE8076,148
466 POKE8077,149:POKE8099,150:POKE8121,151
467 POKE38884,0:POKE38862,0:POKE38840,5:POKE38841,5:POKE38818,5:POKE38819,5
468 POKE38796,5:POKE38797,5
470 FORI=8164T08185:POKEI,147:POKEI+30720,5:NEXT
480 POKE7742,81:POKE7764,66:POKE7763,78:POKE7765,77:POKE7741,67:POKE7743,67
482 POKE7719,77:POKE7720,66:POKE7721,78
490 REM *SPRING*
500 POKE8143,141:POKE8144,141:POKE38863,2:POKE38864,2
510 REM *TUB*
520 POKE8162,143:POKE38882,4:BX=8161:BX=7680
600 AX=INT(RND(1)*12)
610 AX=AX*22+7789
700 FORI=3T022
702 POKEAX+I-3,32:POKEAX+I+20,32:POKEAX+I-24,32:GOSUB5000
730 POKEAX+I-2,128
740 POKEAX+I-1,127
750 POKEAX+I,130
760 ONZXGOTO780
770 POKEAX+I-23,131:ZX=1:GOTO800
780 POKEAX+I+21,132:ZX=0
790 SYS(828):GOSUB5000
795 PRINT"SH";PEEK(1004):"LITERS"
800 NEXT
810 POKEAX+22,32:POKEAX+21,32:POKEAX+20,32:POKEAX+43,32:GOTO6000
1990 REM *MASCHINE-LANGUAGE*
2000 DATA173,232,3,201,0,208,189,165,197,201,37,240,84,201,29,208,71,174,233,3,2
34
2010 DATA234,234,234,234,224,3,234,234,240,59,172,234,3,192,1,240,14,169,138,157
2020 DATA206,31,169,1,141,234,3,76,123,3,234,169,140,157,206,31,169,0,141,234,3,
234
2030 DATA169,137,157,184,31,232,169,32,157,184,31,157,206,31,202,202,169,139,157
,206
2040 DATA31,142,233,3,234,96,234,169,1,141,235,3,96,234,174,233,3,232,234,234,23
4,234
2050 DATA224,19,234,234,240,67,172,234,3,192,1,240,14,169,134,157,206,31,169,1,1
41
2060 DATA234,3,76,203,3,234,169,136,157,206,31,169,0,141,234,3,234,169,133,157,1
84
2070 DATA31,202,169,32,157,184,31,157,206,31,232,234,169,135,157,207,31,142,233,
3
2080 DATA234,96,234,234,234,234,18,0,0,0,234,173,235,3,201,1,208,9,238,236,3,169
,0
2090 DATA141,235,3,234,96
4990 REM *DROPPING*
5000 POKEBX-22,32

```

```

5005 IFA1%=1THEN5100
5010 AA%=INT(RND(1)*4):IFAA%>2THENRETURN
5020 A1%=1:BB%=AA+I+21
5100 IFPEEK(BB%+22)>32THENIFPEEK(BB%+22)<141THEN6000
5103 IFBB%>8142THENA1%=0:POKE36875,0:RETURN
5105 TT%=(8142-BB%)/5+128:POKE36875,TT%:POKEBB%,142:BB%=BB%+22:RETURN
5990 REM *HIT*
6000 PP%=PEEK(1001):POKE8142+PP%,145:POKE8141+PP%,144:POKE8143+PP%,146
6019 POKE8120+PP%,32:POKE8121+PP%,32
6020 POKE36875,0:FORI=255TO128STEP-6:POKE36877,I:NEXT:POKE36877,0
6030 PRINT"AGAIN (Y/N)"
6040 GETA$:IFA$=""THEN6040
6050 IFA$="Y"THENRUN320
6052 IFA$="N"THEN6060
6054 GOTO6040
6060 PRINT"J":END

```

Motodrom

8K RAM

MOTODROM

In this game you drive a car on a famous race track.
Your car can be controlled by a joystick.

At the beginning you have to enter the number of
drivers and their names, as well as the speed each
driver dares to drive.

Each crash causes a delay, too many crashes cause a
disqualification. The driver with the best time
wins.

```

0 POKE36878,15:RUN 100
1 P=P+21:RETURN
2 P=P+22:RETURN
3 P=P+23:RETURN
4 P=P-1:RETURN
5 P=P:RETURN
6 P=P+1:RETURN
7 P=P-23:RETURN
8 P=P-22:RETURN
9 P=P-21:RETURN
20 REM

```



```

440 PRINTINT((TI-T)/60)"SECONDS";
446 O(G)=0
450 FORI=1TO6000:NEXT
460 IFG<HGOTO152
470 GOTO900
800 PRINT"*****M O T O D R O M*****"
803 GOTO1200
804 FORG=1TOH
805 PRINT"NAME OF DRIVER # "G:INPUTF$(G)
806 NEXT
807 FORI=0TO2000:G=0:G2=0
820 RETURN
900 PRINT"□"
910 PRINT"*****SCORE AFTER 1ST RUN ";
915 PRINT"***** IN FORMULA 1 *****"
920 PRINT"*****"
925 PRINT"DRIVER";" SECONDS*****"
930 FORI=1TOH
931 IFT1>=90THENO(I)=999
932 PRINTF$(I),O(I)" "
934 NEXT
955 PRINT"*****"
960 INPUT"AGAIN ";A$
970 IFA$="Y"THENG=0:GOTO152
980 IFA$="N"THENEND
990 PRINT"□";:GOTO960
1200 PRINT"*****WELCOME TO THE 1ST RUN IN "
1210 PRINT"FORMULA 1 IN MONTE CARLO"
1255 PRINT:PRINT"*****PRESS ANY KEY*****"
1256 GETG$:IFG$=""THEN1256
1257 PRINT"□"
1260 PRINT"*****HERE IS THE ROUTE. YOU HAVE TO"
1270 PRINT"DRIVE THROUGH IT TWO TIMES"
1280 PRINT"      !                ←      "
1290 PRINT"      ┌───┐ ┌───┐ ┌───┐ ┌───┐ "
1300 PRINT"      │   │ │   │ │   │ │   │ "
1310 PRINT"      Z │   │ ┌───┐ ┌───┐ │   │ "
1320 PRINT"      ST │   │ │   │ │   │ │   │ "
1330 PRINT"      │   │ │   │ │   │ │   │ "
1340 PRINT"      └───┘ └───┘ └───┘ └───┘ "
1345 PRINT:PRINT"*****PRESS ANY KEY*****"
1346 GETG$:IFG$=""THEN1346
1347 PRINT"□"

```

```

1350 PRINT"WATCH WATCH OUT FOR THE DANGEROUS PARTS"
1360 PRINT"MARKED (!). "
1380 PRINT"ARE YOU READY"
1390 GETQ$:IFQ$=""THENGOTO1390
1391 IFQ$="Y"THENGOTO1400
1392 GOTO1390
1400 PRINT"Q":GOTO1470
1470 PRINT"ENTER NUMBER OF DRIVERS ":INPUTH
1500 GOTO804

```

Fill the aisles

8K RAM

FILL THE AISLES

In this game you fill aisles by moving around your player. A second player (controlled by the computer) moves around also and clears the aisles. Try to fill as many aisles as possible and avoid getting hit by the other player. You will get a time-bonus for a good score.

```

0 DIMX(255)
1 DIME$(11),HN$(11)
2 POKE36879,15
5 PRINT"Q":POKE36879,93:PRINT"WATCH WATCH OUT FOR THE DANGEROUS PARTS"
6 FORI=1TO3000:NEXT:POKE36879,27
10 PRINT"QJOYSTICK HOOKED UP?"
11 GETA$:IFA$=""THEN11
12 IFA$="Y"THENPOKE930,1:GOTO30
13 IFA$="N"THENPOKE930,0:GOTO30
14 PRINT"Y OR N PLEASE ":GOTO10
30 FORI=828TO924
35 READA:POKEI,A:NEXT
40 DATA173,162,3,201,1,208,3,76,76,3,165,197,141,163,3,96
45 DATA169,0,141,19,145,169,127,141,34,145,173,32,145,41,128,201
50 DATA0,208,6,169,37,141,163,3,96
55 DATA169,255,141,34,145,173,17,145,41,0,201,0,208,6,169,17,141,163,3,96
60 DATA173,17,145,41,16,201,0,208,6,169,29,141,163,3,96
65 DATA173,17,145,41,4,201,0,208,6,169,48,141,163,3,96
70 DATA169,10,141,163,3,96
80 GOSUB50000
85 BO=0
90 PRINT"Q"
100 PRINT"Q"
200 FORI=1TO6
250 PRINT"Q"
275 PRINT"Q"
276 PRINT"Q"
280 NEXT

```



```

290 PRINT"  "
300 PRINT"  "
400 XX=80
500 PU=0
1000 AN=7725
1005 SP=8139
1007 X(48)=-22:X(37)=1:X(17)=22:X(29)=-1
1010 CR=22
1015 TI#="000000"
1020 LC=AN
1030 AN=AN+CR:IFPEEK(AN)=160THENAN=AN-CR:GOTO10010
1035 IFSP=ANTHEN20100
1040 ZR=INT(20*RND(1)):IFZR=6THENAN=AN-CR:GOTO10010
2000 POKEAN,42:POKELC,32
3020 PS=SP:SYS(828):R=PEEK(931):SP=SP+X(R):IFPEEK(SP)=160THENSP=SP-X(R)
3030 POKESP+30720,5:POKESP,81
3040 IFPS<SPTHENPOKE36875,200:POKEPS+30720,0:POKEPS,46:POKE36875,0
3900 IFSP=ANTHEN20100
4000 VZ=INT((XX-(TI/60))):PRINT"  ";VZ"  ":IFVZ<=0THENGOTO20000
9999 GOTO1020
10000 REM CONTROL BY COMPUTER
10005 IFSP=ANTHEN20100
10010 RA=INT(4*RND(1))
10020 IFRA=0THENCR=1
10030 IFRA=1THENCR=-1
10040 IFRA=2THENCR=22
10050 IFRA=3THENCR=-22
10060 GOTO1030
20000 FORI=1TO500:NEXT
20005 IFB0=1THENPRINT"  "
20050 POKEAN,32:POKESP,32
20060 FORP=7725TO8139
20070 IFPEEK(P)=46THENPU=PU+1:POKE36876,128+INT(PU/2.8):FORI=1TO20:NEXT:POKE368
76,0
20075 IFPEEK(P)=46THENPRINT"  ";PU"  ":POKEP,32
20080 NEXT:GOTO20700
20100 FORI=1TO5
20110 POKE36877,200:POKESP,81
20120 FORJ=1TO50:NEXTJ
20130 POKE36877,0:POKESP,96
20140 FORJ=1TO50:NEXTJ:NEXTI:POKEAN,96:POKEAN-CR,96:GOTO20000
20700 B0=B0+1:IFB0=1THENPRINT"  "
21000 FORI=1TO500:GETS#:NEXT
21010 PRINT"YOUR SCORE="PU
21020 INPUT"YOUR NAME":N#
21030 E$(11)=STR$(PU)+" PUNKTE="+N#
21040 FORI=1TO10:E$(I)=HN$(I):NEXT
21050 GOSUB60000
21060 PRINT"  "
21116 PRINT
21120 FORI=1TO10
21125 IFI<10THENPRINT"  ";
21130 PRINTI".PLACE":HN$(I):NEXT
30040 PRINT"  "
30050 GETA#:IFA#="Y"THEN80
30060 IFA#<>"N"THEN30050
30070 END
50000 PRINT"  "
50010 GETA#:IFA#=""THEN50010
50020 IFA#<>"Y"THENRETURN
50030 PRINT"  "
50040 PRINT"TRY TO FILL THE AISLES"
50050 PRINT"  "
50055 PRINT"PLAYER"
50080 PRINT"CONTROL YOUR PLAYER WITH THE KEYS"
50090 PRINT"  "
50100 PRINT"AND  "
50107 PRINT:PRINT:PRINT"PRESS ANY KEY !"
50108 GETA#:IFA#=""THEN50108
50109 PRINT"  "
50110 PRINT"YOUR PLAYER ONLY MOVES, IF THE"
50120 PRINT"PROPER KEY IS PRESSED."
50130 PRINT"EACH SQUARE THAT YOU PASS WILL"
50140 PRINT"BE MARKED BY A DOT."
50150 PRINT"THE COMPUTER ALSO MOVES A"
50160 PRINT"FIGUR (*).THIS FIGUR DELETES THOSE DOTS"
50170 PRINT"PLACED BY YOU."

```

```

50172 PRINT:PRINT:PRINT:"PRESS ANY KEY :■
50173 GETA$:IFA$=""THEN50173
50174 PRINT"□"
50180 PRINT"COLLISIONS WITH THE PLAYER";
50185 PRINT"OF THE COMPUTER FINISH
50187 PRINT"THE GAME.
50200 PRINT"AFTER 80 SECONDS THE FIRST ROUND
50210 PRINT"IS OVER AND THE COMPUTER COUNTS
50220 PRINT"THE DOTS PLACED BY YOU.FOR EVERY
50230 PRINT"10 POINTS YOU GET
50235 PRINT"AN ADDITIONAL
50240 PRINT"1 SECOND PLAYTIME.
50250 PRINT"
50260 GETA$:IFA$=""THEN50260
50270 PRINT"□"
51010 PRINT"XXXXXXXXXXREADY
51020 FORI=1TO1000:NEXT
51030 PRINT"STEADY
51040 FORI=1TO1000:NEXT
51045 PRINT
51050 FORI=1TO10:PRINT"□□□□":FORJ=1TO50:NEXTJ:PRINT"□□□□":FORJ=1TO50:NEXTJ:NEXT
TI
52000 RETURN
60000 Q1=11
60010 Q2=Q1/2
60020 Q1=INT(Q2)
60030 IFQ1=0THEN61000
60040 Q3=11-Q1
60050 Q4=1
60060 Q5=Q4
60070 Q6=Q5+Q1
60080 IFE$(Q5)=E$(Q6)THEN60140
60090 T$=E$(Q5)
60100 E$(Q5)=E$(Q6)
60110 E$(Q6)=T$
60120 Q5=Q5-Q1
60130 IFQ5=1THEN60070
60140 Q4=Q4+1
60150 IFQ4>Q3THEN60010
60160 GOTO60060
61000 FORI=1TO10:HN$(I)=E$(I):NEXT:RETURN

```

Simple monitor

3.5 k RAM

SIMPLE MONITOR PROGRAM FOR VIC-20

The following program is a simple machine language monitor for your VIC-20. The program works on the standard version without memory expansion and gives you the following options:

C = Change memory location
M = Dump memory location
G = Start of a machine-language program
S = Save a machine-language program
R = Read a machine-language program

This very simple monitor program is meant for the first contact with machine-language and allows you to do little expeditions in the RAM/ROM area. The memory map of the VIC-20 (standard version) looks like the following:

(See page 34)

From 0000 hex to 03FFF are the BASIC-vectors and addresses that the VIC-20 internally needs. 0400 hex to 0FFF is empty. (On the PET and CBM the BASIC area starts at 0400 hex).

In the VIC-20 there are about 3.5k of RAM, starting at 1000 hex, available for the user. The upper part of the 4K block is used as the screen memory (1E00-1FFF).

After you have loaded the program there are about 750 bytes left. With the C-command you can look into memory locations and change the contents. After you have entered the start address you can increase the address by pressing the space-bar or lower the address by pressing the key with the up-arrow.

It is important to enter all addresses four-digit in hex. To leave the C-command press RETURN.

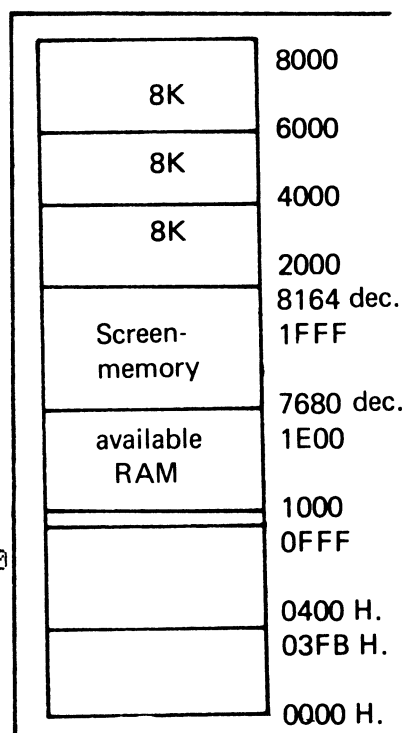
When you change memory locations you have to be careful not to change the BASIC program or the screen memory. You can use the area around 1D00.

The program also allows you to save and load machine-language programs. To save a program you have to enter the start address and the end address and the name.

```

8010 PRINT"?"
8020 PRINT"C=CHANGE MEMORY LOCATION"
8030 PRINT"M=DUMP MEMORY LOCATION"
8040 PRINT"G=START MACHINE LANGUAGE PROGRAM"
8060 PRINT"S=SAVE MACHINE LANGUAGE PROGRAM"
8070 PRINT"R=READ MACHINE LANGUAGE PROGRAM"
8080 PRINT"RETURN=RETURN TO BASIC"
8090 PRINT
8091 PRINT"ENTER OPTION"
8095 PRINT
8096 PRINT"* ";
8110 GET C$:IF C$="" GOTO 8110
8120 PRINTC$
8130 IF C$="C" GOTO 8400
8140 IF C$="M" GOTO 8700
8150 IF C$="G" GOTO 9300
8170 IF C$="S" GOTO 9700
8180 IF C$="R" GOTO 9900
8190 IF ASC(C$)=13 THEN END
8200 PRINT"COMMAND UNKNOWN"
8210 PRINT
8220 PRINT
8230 GOTO 8096
8400 REM
8410 PRINT"START ADDRESS";
8420 INPUT HEX4$
8430 GOSUB 9000
8440 S=DEC4+1
8450 PRINT
8460 S=S-1
8470 GOSUB 9100
8480 PRINT" ";
8490 S0=PEEK(S)
8500 N=3:GOSUB 9150
8510 PRINT" - ";
8520 DEC2=0
8530 FOR I=2 TO 1 STEP-1
8540 GET C$:IF C$="" GOTO 8540
8550 IF C$="↑" GOTO 8450
8560 IF C$=" " GOTO 8640
8570 TEMP=ASC(C$)
8580 IF TEMP=13 GOTO 8095
8590 PRINTC$;

```



```

8600 F=48:IF TEMP>58 THEN F=55
8610 DEC2=DEC2+(TEMP-F)*I*I*I*I
8620 NEXT I
8630 POKE S,DEC2
8640 S=S+1
8650 PRINT
8660 GOTO 8470
8700 REM
8705 PRINT
8710 PRINT"START ADDRESS";
8720 INPUT HEX4$
8730 GOSUB 9000
8740 L=DEC4
8745 PRINT
8750 PRINT"END ADDRESS";
8760 INPUT HEX4$
8765 PRINT
8770 GOSUB 9000
8780 H=DEC4
8790 D=H-L
8800 FOR I=1 TO D STEP 8
8810 S=L-1+I
8820 GOSUB 9100
8830 PRINT"  ";
8840 FOR J=1 TO 8
8850 S0=PEEK(S-1+J)
8860 N=3:GOSUB 9150
8870 PRINT"  ";
8880 NEXT J
8890 PRINT" "
8900 NEXT I
8910 GOTO 8095
9000 REM
9010 DEC4=0:MULT=4096
9020 FOR I=1 TO 4
9030 TEMP=ASC(MID$(HEX4$,I,1))
9040 F=48:IF TEMP>58 THEN F=55
9050 DEC4=DEC4+(TEMP-F)*MULT
9060 MULT=MULT/16
9070 NEXT I
9080 RETURN
9100 REM
9110 N=1

```

```

9120 S(1)=INT(S/4096)
9130 S(2)=INT((S-S(1)*4096)/256)
9140 S0=(S-S(1)*4096)-S(2)*256
9150 S(3)=INT(S0/16)
9160 S(4)=(S0-S(3)*16)
9170 FOR K=N TO 4
9180 F=48:IF S(K)>9 THEN F=55
9190 PRINTCHR$(S(K)+F);
9200 NEXT K
9210 RETURN
9300 REM
9305 PRINT
9310 PRINT"START ADDRESS";
9320 INPUT HEX4$
9330 GOSUB 9000
9340 SYS(DEC4)
9350 GOTO 8095
9700 REM
9705 PRINT
9710 PRINT"START ADDRESS";
9720 INPUT HEX4$
9730 GOSUB 9000
9740 L=DEC4
9745 PRINT
9750 PRINT"END ADDRESS";
9760 INPUT HEX4$
9770 GOSUB 9000
9780 H=DEC4
9790 D=H-L
9791 PRINT
9792 FS=-1:FD=-1:FC=191
9793 PRINT"FILENAME";
9794 INPUT F$
9795 OPEN1,1,1,F$
9797 J=LEN(STR$(D))
9799 FS=FS+J
9810 PRINT#1,D
9820 FOR J=0 TO D
9830 I=PEEK(L+J)
9840 PRINT#1,I
9841 K=LEN(STR$(I))
9842 FE=INT(-(FS+K)/FC)
9843 IF FE=FD GOTO 9849

```

```

9844 POKE 59411,53
9845 T=TI
9846 IF(TI-T)<5 GOTO 9846
9847 POKE 59411,61
9848 FS=-1
9849 FS=FS+K
9850 NEXT J
9860 CLOSE 1
9865 PRINT
9870 PRINT"SAVED"
9880 GOTO 8095
9900 REM
9905 PRINT
9906 PRINT"FILENAME";
9907 INPUT F$
9908 PRINT
9910 PRINT"REWIND AND PRESS ANY KEY"
9920 GET A$: IF A$="" THEN 9920
9930 OPEN1,1,0,F$
9940 INPUT#1,D
9945 PRINT"D"
9950 PRINT"START ADDRESS";
9960 INPUT HEX4$
9970 GOSUB 9000
9974 PRINT
9975 L=DEC4
9980 FOR I=0 TO D
9990 INPUT#1,X
9991 PRINT"X";
9992 POKE L+I,X
9993 NEXT I
9994 CLOSE1
9995 PRINT:PRINT
9996 PRINT"END OF FILE"
9997 GOTO8095

```

Programming in machine-language

PROGRAMMING IN MACHINE-LANGUAGE

In this chapter we will show you how to program in machine language with your VIC-20.

If you do I/O programming, you will find out that for some applications BASIC is too slow and you have to use a machine language routine.

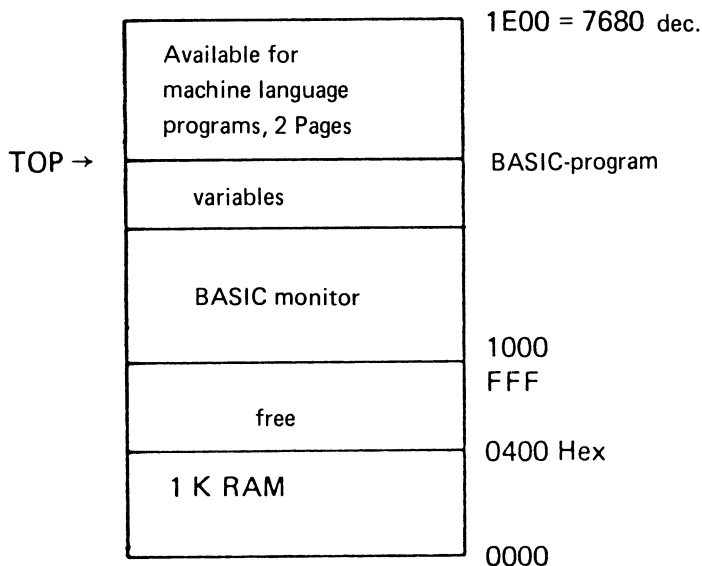
We will show you the basics here and try to stimulate your interest.

The VIC-20 works with a 6502 microprocessor. The following tables show you all the commands of that microprocessor. To enter the commands you can use the machine language monitor previously used.

Here is an example:

Enter the monitor program mentioned above and test it. Now enter PRINT FRE(0). You should get 986 bytes free. This means we can use about two pages = 510 bytes for our experiments. The memory map now looks as follows:

(4330 bytes free on a 8K VIC)



Caution: If you use a 3k memory expansion between 0400 and 0FFF hex or an expansion above 2000 hex the memory map looks different !

By ?PEEK(55) and ?PEEK(56) we get the top of memory.

(55) = 0 dec = 00 hex

(56) = 30 dec = 1E hex

Since the first number is the lower byte, and the second number is the higher byte of the address, we get 1E00 hex or 7680 dec for top of memory.

To protect the upper portion from BASIC, in order to place a machine language program there, we have to change the boundary by two POKE commands in locations 55 and 56.

We will redefine the top of BASIC to 1C00 = 7168 dec by POKE 55,0 and POKE 56,28. We then will have 425 bytes left for BASIC variables and 2 pages = 510 bytes for our machine language program.

Commands	symb. Code	Operation	Adressing modes												condition codes								
			IMM.	ABS	ABS,X	ABS,Y	Z0	Z0,X	Z0,Y	(IND,X)	(IND),Y	REL	IND	ACCU	IMPL	N	Z	C	1	D	V		
			A9 A2 A0	AD AE AC	BD BE BC	89 8E 8C	A5 A6 A4	B5 B6 B4		A1 86 81	B1 86 91												
Transport	LDA	M → A																X	X	-	-	-	
	LDX	M → X																X	X	-	-	-	
	LDY	M → Y																X	X	-	-	-	
	STA	A → M																-	-	-	-	-	
	STX	X → M																-	-	-	-	-	
	STY	Y → M																-	-	-	-	-	
	TAX	A → X																-	-	-	-	-	
	TAY	A → Y																-	-	-	-	-	
	TXA	X → A														AA	X	X	-	-	-		
	TYA	Y → A														AB	X	X	-	-	-		
	TXS	X → S														8A	X	X	-	-	-		
	TSX	S → X														98	X	X	-	-	-		
	PLA	S+1 → S, Ms → A														9A	X	-	-	-	-		
	PHA	A → Ms, S-1 → S														BA	X	X	-	-	-		
	PLP	S+1 → S, Ms → P														68	X	X	-	-	-		
	PHP	P → Ms, S-1 → S														48	-	-	-	-	-		
																28	-	-	-	-	-		
															08	-	-	-	-	-			
arithmetic-	ADC	A+M+C → A	69	6D	7D	79	65	75		61	71							X	X	X	-	-	X
	SBC	A-M-C → A	E9	ED	FD	F9	E5	F5		E1	F1							X	X	X	-	-	X
	INC	M+1 → M		EE	FE		E6	F6										X	X	-	-	-	
	DEC	M-1 → M		CE	DE		C6	D6										X	X	-	-	-	
	INX	X+1 → X														E8	X	X	-	-	-		
	DEX	X-1 → X														CA	X	X	-	-	-		
	INY	Y+1 → Y														CB	X	X	-	-	-		
	DEY	Y-1 → Y														88	X	X	-	-	-		
logic-	AND	A ∧ M → A	29	2D	3D	39	25	35		21	31							X	X	-	-	-	
	ORA	A ∨ M → A	09	0D	1D	19	05	15		01	11							X	X	-	-	-	
	EOR	A ⊕ M → A	49	4D	5D	59	45	55		41	51							X	X	-	-	-	
compare-	CMP	A-M	C9	CD	DD	D9	C5	D5		C1	D1							X	X	X	-	-	-
	CPX	X-M	E0	EC			E4											X	X	X	-	-	-
	CPY	Y-M	C0	CC			C4											X	X	X	-	-	-
	BIT	A ∧ M		2C			24											7	X	-	-	-	6
branch-	BCC	BRANCH ON C=0										90						-	-	-	-	-	
	BCS	BRANCH ON C=1										80						-	-	-	-	-	
	BEQ	BRANCH ON Z=1										F0						-	-	-	-	-	
	BNE	BRANCH ON Z=0										D0						-	-	-	-	-	
	BMI	BRANCH ON N=1										30						-	-	-	-	-	
	BPL	BRANCH ON N=0										10						-	-	-	-	-	
	BVC	BRANCH ON V=0										50						-	-	-	-	-	
	BVS	BRANCH ON V=1										70						-	-	-	-	-	
	JMP																	-	-	-	-	-	
	JSR			4C	20								6C					-	-	-	-	-	
SHIFT-	ASL			0E	1E		06	16							0A		X	X	X	-	-	-	
	LSR			4E	5E		46	56							4A		0	X	X	-	-	-	
	ROL			2E	3E		26	36							2A		X	X	X	-	-	-	
	ROR			6E	7E		66	76							6A		X	X	X	-	-	-	
Status- Register	CLC	C=0														18	-	-	0	-	-	-	
	CLD	D=0														D8	-	-	-	-	0	-	
	CLI	I=0														58	-	-	-	0	-	-	
	CLV	V=0														B8	-	-	-	-	0	-	
	SEC	C=1														38	-	-	1	-	-	-	
	SED	D=1														F8	-	-	-	1	-	-	
	SEI	I=1														78	-	-	-	1	-	-	
Misc.	NOP	NO OPER														EA	-	-	-	-	-	-	
	RTS	RETURN F. SUB														60	-	-	-	-	-	-	
	RTI	RETURN F. INT														40	-	-	-	-	-	-	
	BRK	BREAK														00	-	-	-	1	-	-	

Table I

Now we enter the following program using the command C of the machine language monitor:

		ORG \$1C10
1C10:	A900	START LDA #\$00
1C12:	A209	LDX #\$09
1C14:	9DFF1C	LOOP STA \$1CFF,X
1C17:	CA	DEX
1C18:	10FA	BPL LOOP
1C1A:	60	RTS

After you have entered the program, look at some locations from 1CFF. They should contain FF. Our machine language program will write nulls to these locations.

Start the program using the command G1C10. The command RTS at location 1C1A returns us to our previous location. Now enter C and the address 1CFF. Press the space bar several times and you will see that locations 1CFF to 1D08 now contain 00.

Now that you have written and tested your first machine language program let's have a closer look at it. First we loaded the accumulator with 00: LDA #0. Then we loaded the X-register with 09: LDX #\$09. Next we put the nulls from the accumulator to 1CFF + 9 = 1D08: STA \$1CFF,X. This command uses indexed addressing, which means that the effective address is an absolute address (1CFF), plus the contents of an index register (9).

The next command decrements the X-register: DEX. This means that the next effective address is 1D07. Now the X-register is decremented to zero. This way the last effective address is 1CFF+0=1CFF. The X-register is then decremented again and contains FF which is negative and thus the BPL-command (branch if plus) is no longer executed.

The value next to the BPL-command defines the relative distance to jump (backwards or forwards). In our example the program should branch backwards:

F9
FA = Jump back six locations
FB
FC
FD
FE
FF = Jump back one location

From lC1A to lC14 it is six locations backwards, so we have to place the value FA behind the BPL-command.

The RTS (return from subroutine) brings us back to BASIC.

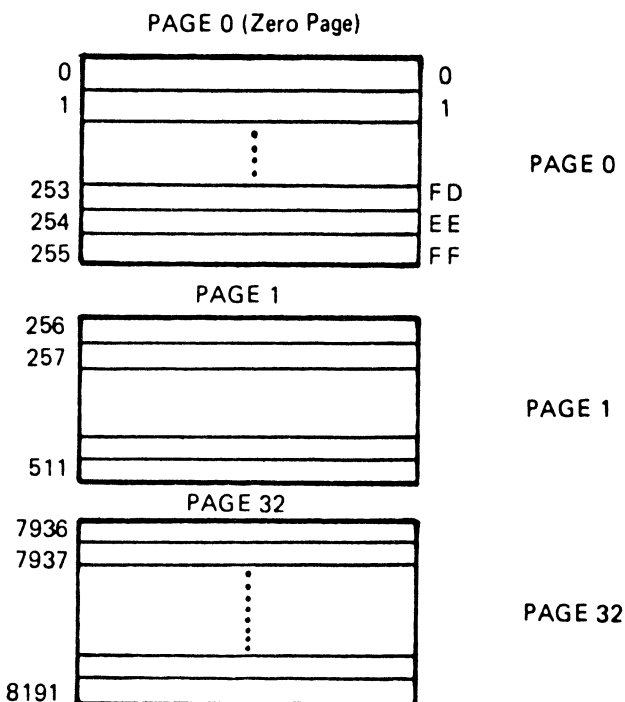
Here are two suggestions:

- 1.) Change the program so that it writes 44 in locations lC10 to lD08.
- 2.) Change the program so that it writes 1 to locations which belong to the 3.5K working memory of the VIC-20 (1000-1BFF).

As you already know you have to protect your machine language program from being overwritten from BASIC. Location 56 contains an address that defines the memory available. Check with ?PEEK (56). You will get the number 28, which means that the memory of your VIC-20 ends at page 28. 28 pages with 256 locations in each is 7168 words minus the empty area from 0400 to 0FFF and minus the RAM area used by the system makes 3581 bytes of free RAM.

If you turn on your VIC-20 it shows you a lower value. The reason for this is that a small part of the RAM has been reserved for saving intermediate values.

Caution! If you save the machine language program on cassette with the monitor-program and want to load it later you have to define the top of memory with POKE 56,28. The value 28 remains at location 56 until you turn the computer off or you poke another value.



MACHINE - LANGUAGE ROUTINES IN A BASIC PROGRAM

In this chapter we will show you a BASIC program that translates a machine-language program into DATA-statements. The program is called POKER MAKER. POKER MAKER generates a program that pokes decimal values into memory. You have to enter start and end address.

Caution! Due to self destruction, you must save the program prior to program implimentation.

```

502 PRINT:PRINT"THIS PROGRAM GENERATES"
504 PRINT"A BASIC PROGRAM
506 REM THIS PROGRAM GENERATES A MACHINE
507 REM LANGUAGE PROGRAM IN DATA STATEMENTS
508 REM WHICH CAN BE POKED IN LATER
510 REM MAKE SURE THAT THERE IS A PROGRAM
511 REM IN RAM.
512 INPUT "START ADDRESS DEC":L1

```

```

514 INPUT "END ADDRESS DEC":L2
516 IF L2<L1 THEN 500
518 PRINT"OK"
520 PRINT"20FOR I=":L1;"TO":L2;" :READDC:POKEI,DC:NEXT I"
522 PRINT"L1=":L1;" :L2=":L2;" :GOTO528"
524 POKE 198,10:FOR N=0 TO 9:POKE 631+N,13:NEXT N
526 PRINT"OK":END
528 ML=L1:LN=22
530 PRINT"OK"
532 PRINT LN:"DATA"
534 D1=PEEK(ML):D1$=STR$(D1)
536 DC$=RIGHT$(D1$,LEN(D1$)-1)
538 PRINT DC$:ML=ML+1
540 IF POS(0)<18 THEN 544
542 PRINT:PRINT"LN=":LN;" +2:ML=":ML;" :L2=":L2;" :GOTO 530":GOTO 548
544 IF ML>L2 THEN PRINT:PRINT"GOTO554":GOTO 548
546 PRINT CHR$(44):GOTO 534
548 PRINT"OK"
550 POKE 198,10:FOR N=0 TO 9:POKE 631+N,13:NEXT N
552 PRINT "OK":END
554 I=500:J=502:K=504
556 REM
558 REM
560 PRINT"OK":PRINT I:PRINT J:PRINT K
562 PRINT"I=":I+6;" :J=":J+6;" :K=":K+6;" :GOTO560"
564 PRINT"OK":POKE 198,10:FOR N=0 TO 9:POKE 631+N,13:NEXT N

```

After POKER MAKER is finished, you get an error message which you can ignore. Enter LIST and the generated program appears. The following program was created by POKER MAKER.

```

20 FOR I=7184 TO 7194:READ DC:POKE I,DC:NEXT I
22 DATA 169,0,162,9
24 DATA 157,255,28
26 DATA 202,16,250
28 DATA 96
507 REM PROGRAM IN DATA STATEMENTS

```

This program pokes the above machine language program to memory starting at 7184 dec.

Before you start this machine language program, you have to protect it again with POKE 56,28.

The above BASIC program can now be saved on cassette. If you need the program later, load it, RUN it, and start it with SYS (7184). This command executes the machine language program and returns to BASIC after it is finished.

The following short program changes locations 7184 through 7194 and displays them.

```
100 FOR I=7184 TO 7194:POKE I,0:NEXT I
200 FOR I=7184 TO 7194:PRINT PEEK(I):NEXT I
```

The locations that are set to zero by our machine language program are 1CFF hex (7423 dec) through 1D08 hex (7432 dec).

The following program displays these locations:

```
300 FOR M=7423 TO 7432:PRINT PEEK(M):NEXT M
350 END
400 FOR N=7423 TO 7432:POKE N,255:NEXT N
```

RUN 300 will display the contents.

RUN 400 writes in a value (255) in order to check it later.

```
20 FOR I=7184 TO 7194:READ DC:POKE I,DC:NEXT I
22 DATA 169,0,162,9
24 DATA 157,255,28
26 DATA 202,16,250
28 DATA 96
29 END
100 FOR I=7184 TO 7194:POKE I,1:NEXT I
200 FOR I=7184 TO 7194:PRINT PEEK(I):NEXT I
300 FOR M=7423 TO 7432:PRINT PEEK(M):NEXT M
350 END
400 FOR N=7423 TO 7432:POKE N,255:NEXT N
507 REM PROGRAM IN DATA STATEMENTS
```

First we start with RUN 400. Then we start the main program with RUN. Next enter SYS 7184 to start the machine language program. Finally we check with RUN 300. This tells us if the machine language program has worked.

You can do many things with machine language routines. Two good examples would be, fast graphics on the screen and sound generation.

The USR(X) function

THE USR(X) FUNCTION

With the SYS-command you can call a machine language program. The start address is the number in brackets following the SYS-command. This command does not take or deliver values from the machine language program to the BASIC program. If this is necessary, you have to POKE the values to certain memory locations where they can be used by the machine language program. If the machine language program stores some values to certain locations you can get those values in BASIC using the PEEK-command.

The machine language program has to be terminated by the RTS-command (opcode 60 hex). The program will then return to BASIC.

If there are a large number of variables going to or coming from the machine language program, the SYS-command together with PEEK and POKE becomes quite complicated. In this case it is better to use the USR(X) command.

With the USR(X) command you can hand over a value from BASIC to the machine language program (X in our example) and the machine language program delivers a value to the BASIC program (Y in our example).

Example: Y = USR(X)

Before you call the machine language program with the USR(X)-command you have to enter the start-address into locations 1 and 2.

For our machine language program starting at 1C10 hex has to look like the following:

```
POKE 1,16
POKE 2,28
```



```

10 FOR I=7423 TO 7433:POKE I,255:NEXT I
15 END
20 FOR I=7423 TO 7433:PRINT PEEK(I):NEXT I
25 END
50 Y=USR(X)
60 END

```

V A L U E	HEX/DEC CONVERSION (<64K)				
	16 ³	16 ²	16 ¹	16 ⁰	POSITION
0	0	0	0	0	
1	4096	256	16	1	
2	8192	512	32	2	
3	12288	768	48	3	
4	16384	1024	64	4	
5	20480	1280	80	5	
6	24576	1536	96	6	
7	28672	1792	112	7	
8	32768	2048	128	8	
9	36864	2304	144	9	
A	40960	2560	160	10	
B	45056	2816	176	11	
C	49152	3072	192	12	
D	53248	3328	208	13	
E	57344	3584	224	14	
F	61440	3840	240	15	

How to use this conversion table:

Hex to Decimal

For each of the four hex digits locate the corresponding decimal value and accumulate.

Decimal to Hex

Find in the table the largest number which is smaller than the decimal value being converted and note the position and equivalent hex value. Subtract the previous number from the number being converted and locate that hex value and position. Repeat for each remainder until finished.

Note that maximum values are FFFF (hex) and 65535 (dec).

ASCII-table:

		0		1		2		3		4		5		6		7	
S E C O N D D I G I T ↓	0	NUL	0	DLE	16	SP	32	0	48	@	64	P	80	\	96	p	112
	1	SOH	1	DC1	17	!	33	1	49	a	65	Q	81	a	97	q	113
	2	STX	2	DC2	18	"	34	2	50	B	66	R	82	b	98	r	114
	3	ETX	3	DC3	19	£	35	3	51	C	67	S	83	c	99	s	115
	4	EOT	4	DC4	20	\$	36	4	52	D	68	T	84	d	100	t	116
	5	ENQ	5	NAK	21	%	37	5	53	E	69	U	85	e	101	u	117
	6	ACK	6	SYN	22	&	38	6	54	F	70	V	86	f	102	v	118
	7	BEL	7	ETB	23	'	39	7	55	G	71	W	87	g	103	w	119
	8	BS	8	CAN	24	(40	8	56	H	72	X	88	h	104	x	120
	9	HT	9	EM	25)	41	9	57	I	73	Y	89	i	105	y	121
	A	LF	10	SUB	26	*	42		58	J	74	Z	90	j	106	z	122
	B	VT	11	ESC	27	+	43		59	K	75	[91	k	107		123
	C	FF	12	FS	28	,	44	<	60	L	76		92	l	108		124
	D	CR	13	GS	29	-	45	=	61	M	77]	93	m	109		125
	E	SO	14	RS	30	.	46	>	62	N	78	^	94	n	110		126
	F	SI	15	US	31	/	47	?	63	O	79		95	o	111	DEL	127

Powers of two

4096	2 ¹²	1	2 ⁰
8192	2 ¹³	2	2 ¹
16384	2 ¹⁴	4	2 ²
32768	2 ¹⁵	8	2 ³
65536	2 ¹⁶	16	2 ⁴
131072	2 ¹⁷	32	2 ⁵
262144	2 ¹⁸	64	2 ⁶
524288	2 ¹⁹	128	2 ⁷
1048576	2 ²⁰	256	2 ⁸
2097152	2 ²¹	512	2 ⁹
4194304	2 ²²	1024	2 ¹⁰
8388608	2 ²³	2048	2 ¹¹
16777216	2 ²⁴		

Base converting with the VIC-20

3.5 k RAM

BASE CONVERTING WITH THE VIC-20

The following program allows you to convert numbers to and from different bases. The following kinds of numbers are possible:

- hexadecimal
- decimal
- octal
- binary
- octal-split

```
2 DIM B(20)
5 PRINT"0"
6 PRINTTAB(10)"BASE CONVERTER"
8 PRINT"0000"
12 PRINT"I CONVERT NUMBERS"
13 PRINT"INTO DECIMAL,OCTAL,"
14 PRINT"OCTAL-SPLIT,HEX AND BINARY"
20 PRINT"00ENTER BASE,NUMBER"
21 PRINT"AND CONVERSION WANTED"
25 PRINT:PRINT
26 PRINT"OCTAL-SPLIT='S'"
30 PRINT"DECIMAL='D'  OCTAL='O'"
31 PRINT"HEXADECIMAL='H' BINARY='B'"
32 PRINT"000UP TO 7 DIGITS"
33 GOTO 100
35 PRINT "000AGAIN(Y,N)?";
36 GET A$:IF A$="" THEN GOTO 36
37 IF A$="Y" THEN GOTO 120
38 PRINT"0":PRINT"00000BASE CONVERTING":PRINT
```

```

39 POKE 59468,12: END
40 PRINT "ENTER BASE (D,O,S,H,B)?";
42 GET Z$:IF Z$="" THEN GOTO 42
45 PRINTZ$
50 IF Z$="H" THEN 1400
55 PRINT"OK"
60 INPUT "NUMBER TO CONVERT";A
65 PRINT"OK"
70 PRINT "NEW BASE.... (D,O,S,H,B) ? ";
74 GET Y$:IF Y$="" THEN GOTO 74
76 PRINT Y$
80 PRINT"OK"
90 IF Z$="S" THEN 1260
92 IF Z$="D" THEN 490
94 IF Z$="O" THEN 580
96 IF Z$="B" THEN 640
98 GOTO 640
100 PRINT"OK(PRESS ANY KEY)";
110 GET R$:IF R$="" THEN GOTO 110
120 PRINT"OK"
130 PRINTTAB(1)"ELCOMP NUMBERS"
140 PRINT"OK";
150 GOTO 40
490 IF Y$="O" THEN 540
495 IF Y$="H" THEN 1600
500 IF Y$="S" THEN 1135
510 Z=2
520 Y$="BINARY ="
530 GOSUB 800
535 GOTO 35
540 Z=8
545 PRINT
550 Y$="OCTAL ="
560 GOSUB 800
565 GOTO 35
580 Z=8
590 IF Y$="D" THEN 620
595 IFY$="S"THEN GOSUB920:A=G:GOTO1135
596 IFY$="H"THEN GOSUB 920:A=G:GOTO1600
600 GOSUB 920
610 A=G: GOTO 510
620 GOSUB 920
630 GOTO 35

```

```

640 Z=2
670 IF Y$="D" THEN 700
675 IF Y$="S" THEN GOSUB 920:A=G:GOTO 1135
676 IF Y$="H" THEN GOSUB 920:A=G:GOTO 1600
680 GOSUB 920
690 A=G: GOTO 540
700 GOSUB 920
710 GOTO 35
800 N=0
810 N=N+1
830 B=INT(A/Z)
840 C=A-(B*Z)
850 B(N)=INT(C+.5)
860 A=B
870 IF B>0 THEN 810
872 IF Y$="H" THEN 912
875 IF Y$="S" THEN 912
880 PRINT Y$;
885 GOTO 900
890 N=N-1
895 IF N=0 THEN 911
900 PRINT B(N);
910 GOTO 890
911 PRINT
912 RETURN
920 C=1:E=(-1)
925 G=0
930 B=A/C
935 E=E+1
940 IF B<Z THEN 970
950 C=C*10
960 GOTO 930
970 D=INT(A/C)
980 H=(Z+E)*D
1000 D=D*C
1010 A=A-D
1020 G=G+H
1025 C=C/10
1040 E=E-1
1050 IF E=>0 THEN 970
1055 IF Z$="S" THEN 1070
1060 PRINT "DECIMAL =";G
1065 PRINT

```

```

1070 RETURN
1135 PRINT"OKTAL SPLIT =";
1140 B(3)=0: B(2)=0: B(1)=0
1150 B=INT(A/256)
1160 C1=A-(B*256)
1165 X=0
1170 A=B
1180 Z=8
1190 GOSUB 800
1200 N=3
1210 PRINTB(N);
1220 N=N-1
1230 IF N>0 THEN 1210
1240 X=X+1
1245 IF X>1 THEN PRINT: GOTO 35
1250 A=C1: B(3)=0: B(2)=0: B(1)=0: GOTO1180
1260 REM 08
1265 PRINT
1270 C=1000: Z=8
1280 B=INT(A/C)
1290 C1=A-(B*C)
1294 A=B
1300 GOSUB 920
1310 G1=G*256
1320 A=C1
1330 GOSUB 920
1340 G=G+G1
1350 PRINT"DEZIMAL =" :G
1355 A=G
1360 IF Y$="B" THEN PRINT:GOTO 510
1362 IF Y$="H" THEN 1600
1364 IF Y$="O" THEN 540
1366 GOTO 35
1400 PRINT"NEW":INPUT "NUMBER TO CONVERT":X$:PRINT:PRINT
1401 PRINT"NEW BASE(D,O,S,H,B) ? ";
1402 GET Y$:IF Y$="" THEN GOTO 1402
1403 PRINT Y$:PRINT
1405 HL=0
1406 G=0
1407 HJ=LEN(X$)
1410 FOR I=1 TO HJ
1415 XX$=MID$(X$,I,1)
1420 H=ASC(XX$)
1425 HL=HJ-I
1430 IF H<57 THEN 1445
1435 B(I)=(H-55)*(16^HL)
1440 GOTO 1450
1445 B(I)=(H-48)*(16^HL)
1450 NEXT I
1455 FOR I=1 TO HJ
1460 G=G+B(I)
1465 NEXT I
1468 PRINT:PRINT:PRINT
1470 PRINT"DECIMAL =" :G

```

```

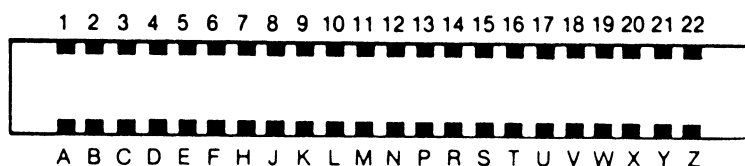
1475 PRINT
1480 A=INT(G+.5)
1485 IF Y$="0" THEN 540
1490 IF Y$="B" THEN 510
1495 IF Y$="S" THEN 1135
1500 GOTO 35
1600 Z=16
1601 PRINT:PRINT"HEXADECIMAL= ";
1610 GOSUB 800
1611 GOTO 1630
1615 N=N-1
1625 IF N=0 THEN 1660
1630 IF B(N)>=10 THEN 1650
1635 A1=B(N)+48
1636 PRINT CHR$(A1);
1640 GOTO 1615
1650 A1=B(N)+55
1655 GOTO 1636
1660 PRINT
1661 GOTO 35
1670 END

```

Universal experimenter board

UNIVERSAL EXPERIMENTER BOARD FOR VIC-20

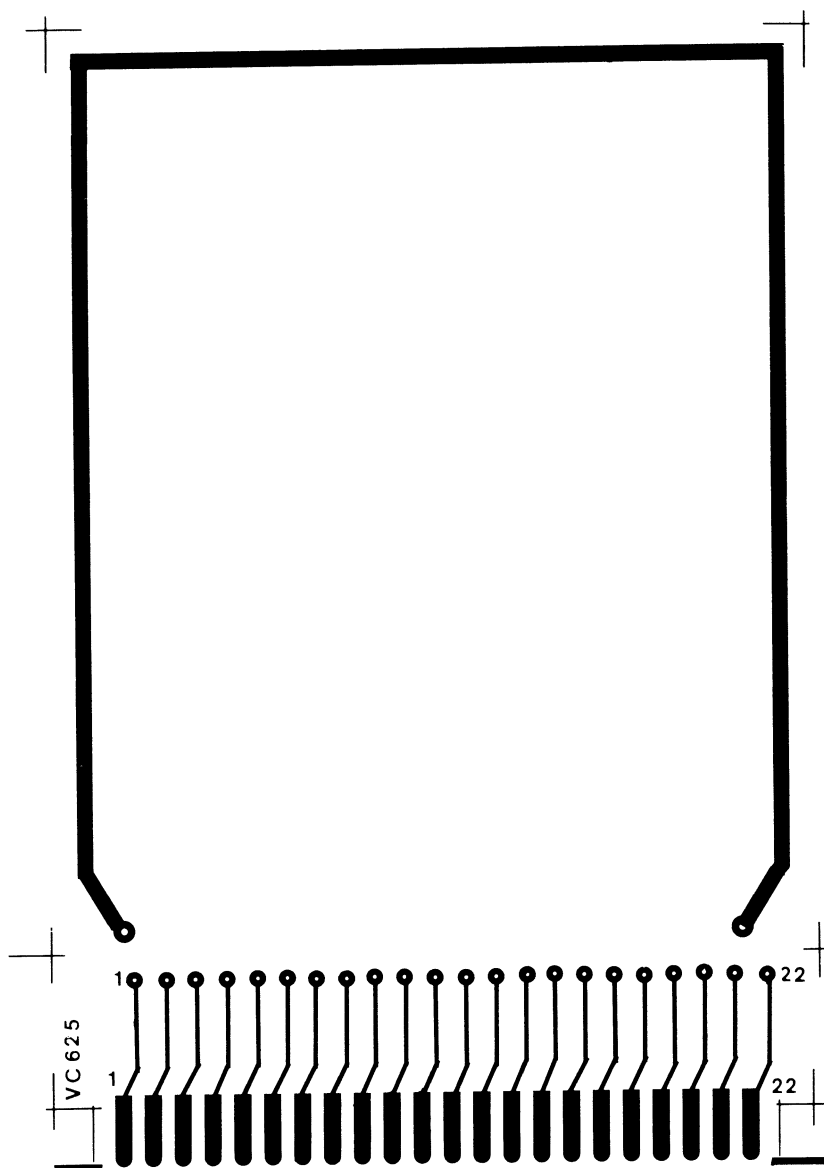
The universal experimenter board gives you the permission to connect external expansions (additional memory, expansion of ports, etc.) directly to the "VIC expansion port". The board has as well as the connector, 2x22 connections which are arranged as follows (seen from the backside):

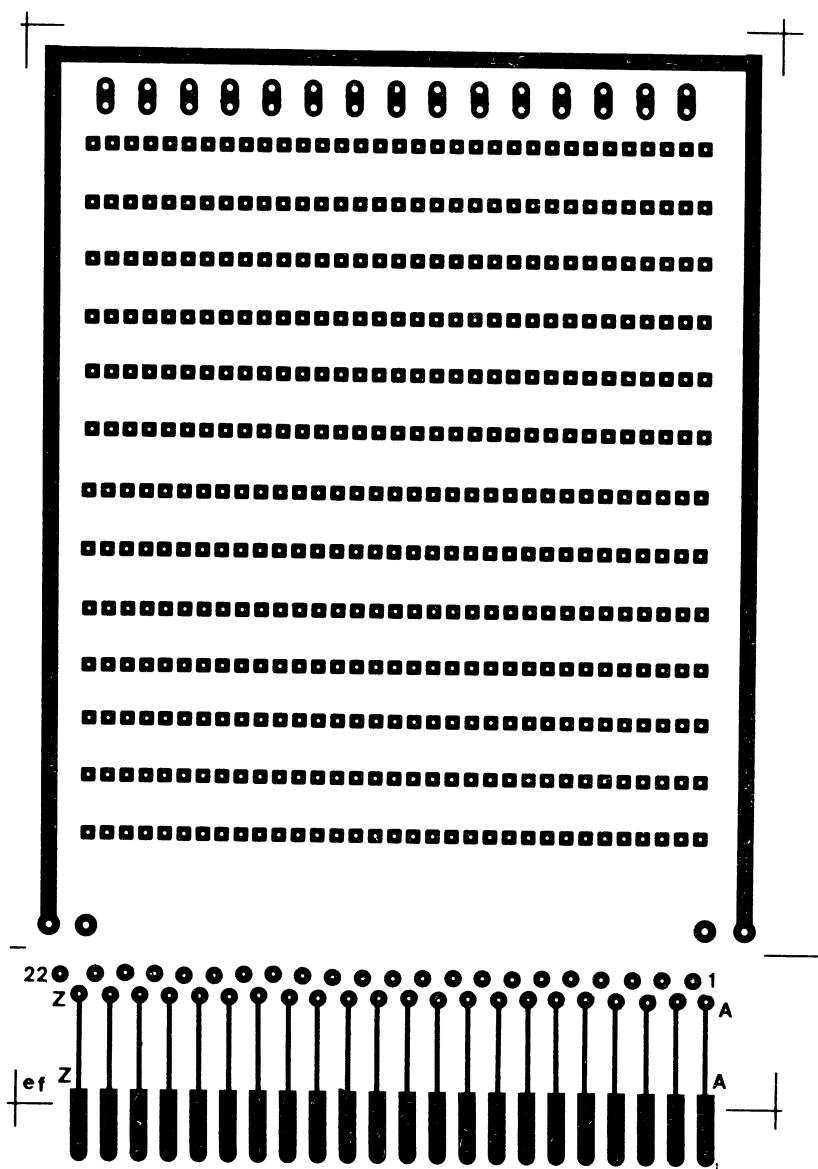


This bus is also called S44 standard bus (standard of the American industry). The only difference is that the naming goes reverse, i.e. A is at the right side.

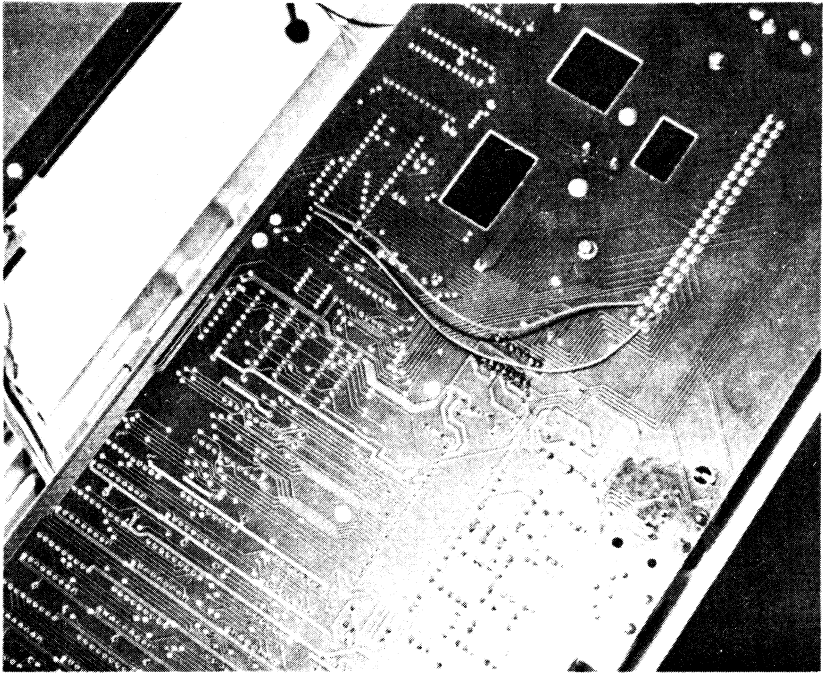
The board has traces on both sides. The signals can easily be brought to the opposite side by using a piece of wire, connecting it through the board. CAUTION! The numbering on the board is opposite to the connector. At the "memory expansion" connector of the VIC-20 there are all data-signals directly from the 6502A CPU and the addresses CA0-CA13.

If needed, you should first connect the two missing address signals A14 and A15 from the CPU to the free pins Y and 20. To do that, open the case and take



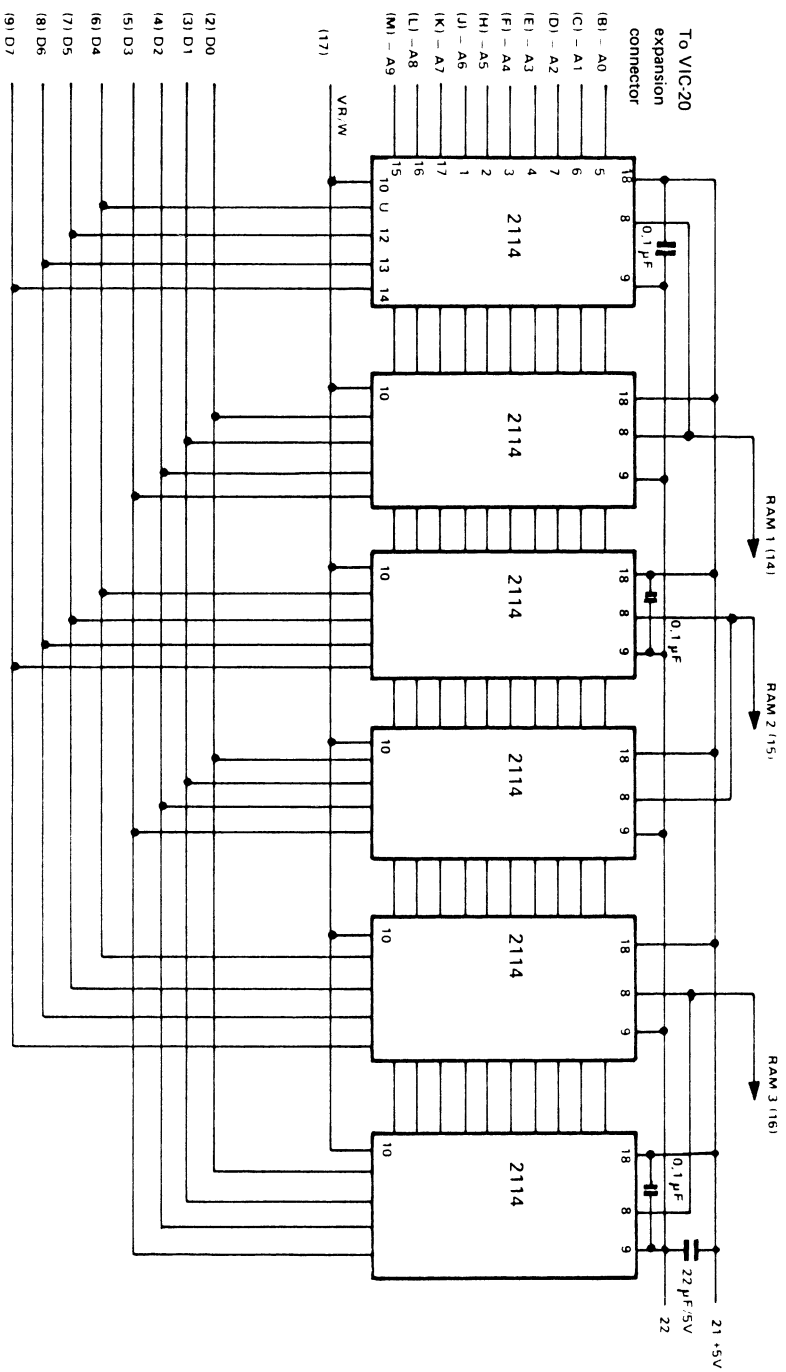


out the motherboard. Then turn the board so that it lies in front of you as in the picture below. # Figure 2



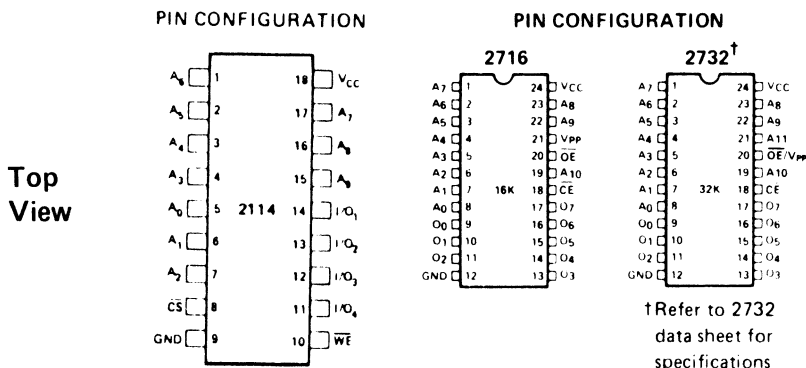
Now connect the two jumpers:
From Y to pin 24 at the CPU
From 20 to pin 25 at the CPU

Besides the address and data signals, there are the signals RAM 1, RAM 2, RAM 3 at the port. These signals are for selecting the three 1k blocks between 400 and FFF hex. That means that a memory expansion of 3k can be assembled without an external decoding. All you need are six memory chips 2114 or L2114, three capacitor 0.1 uF/60V, and six 18 pin DIL sockets.



3K RAM expansion 0400 — 1FFF for the VIC-20

Since the 2114 RAMs contain 1024 x 4 bit, we have to use two chips parallel for a block of 1k x 8 bit. For that we connect DO-D3 to I/O -I/O of the one D4-D7 to I/O - I/O of the other chip.



IMPORTANT - Read this notice

Any warranty that may exist on any computer hardware product that has been modified, altered, or has the seal broken is VOID.

The R/W - signal is to be connected to WE (pin 10) of all chips. The signals RAM 1, RAM 2, RAM 3 select the 1k blocks and go to CS (pin 8) of respectively two 2114 chips.

The power supply for the six RAM chips comes from the VIC-20 also (pins 21,22 at the connector).

For an expansion of more than 3k, you have to use an external decoding. The signals BLK1, BLK2, BLK3, and BLK5 are at the connector (pins 10-13) and can be used for parts of the decoding. A 74LS138 chip (UB1) generates the signals BLK0-BLK7, at which four of them are at the connector.

BLK0 selects the internal 8k block.

BLK1 selects the 2nd 8k block (2000-3FFF).

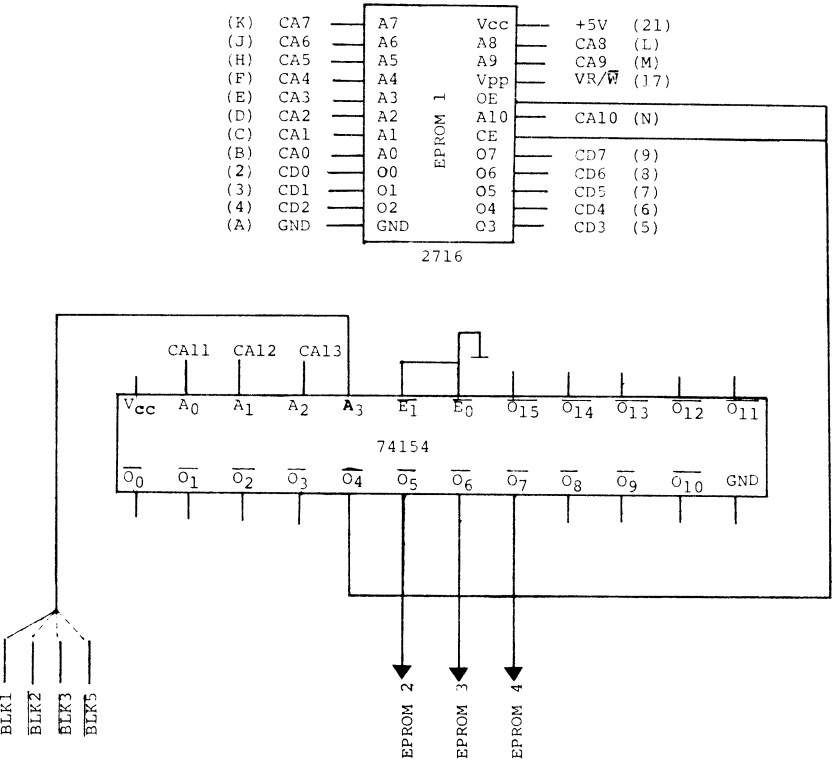
BLK2 selects the 3rd 8k block (4000-5FFF).

BLK3 selects the 4th 8k block (6000-7FFF).

These are the lower 32k.

Starting at 8000 the VIC-20 has its character generator, color register, I/O ports etc. These addresses are selected interval by BLK4 (8000 - 9FFF = 5th 8K block). BLK5 selects the 6th block (A000 - BFFF). This is at the connector and is suitable for the so called auto start ROMs. This means that if a programmed ROM is at the address, the VIC-20 jumps after power-up into that program.

EPROMS:



In a auto start ROM the first cells have to look like the following:

\$A004 \$41

\$A005 \$30

\$A006 \$C3

\$A007 \$C2

\$A008 \$CD

Address \$A000 and \$A001 contains the RESET vector (address, where the program should jump to after power up). Addresses \$A002 and \$A003 contain the NMI vector.

During the auto start the operating system, the ports or the expansion are initialized. This means the first five commands in an autostart program have to look like the following:

JSR \$FD8D ; RAM test

JSR \$FD52 ; Set vectors of operating systems

JSR \$FDF9 ; I/O initialization

JSR \$E518 ; Screen initialization

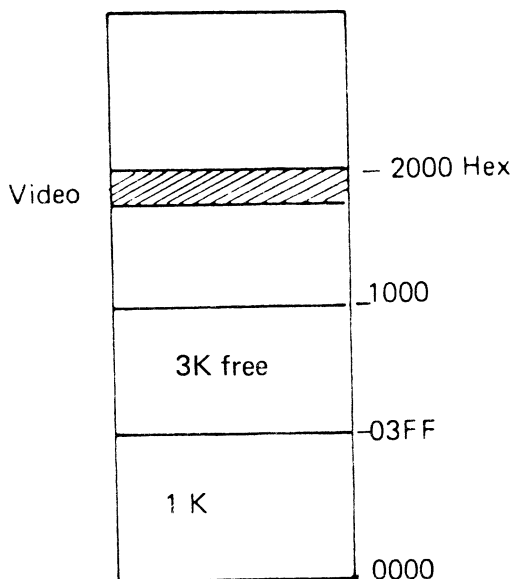
CLI ; Disable interrupts

Memory expansion

MEMORY-EXPANSION FOR VIC-20

The standard version of the VIC-20 comes with 5k of RAM. For many programmers this will become insufficient. For this reason we have developed a 16k RAM/ROM board. This card can hold 1kx8 bit RAMS, and 2kx8 bit EPROMs.

The memory map of the standard version of the VIC-20:



To expand memory up to 8k we have to fill the empty area between 0400 hex and 0FFF hex with 3k of RAM (1 chip 2kx8 bit and 1 chip 1kx8 bit).

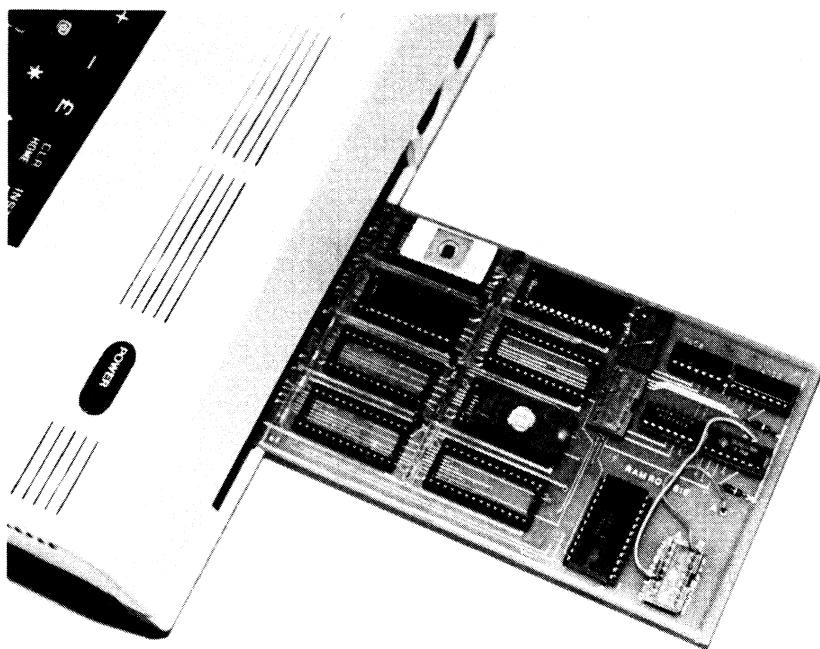
We use block A on the board. For the 3k RAM we need:

1 RAM 1kx8 (4118 Mostek) in socket A1

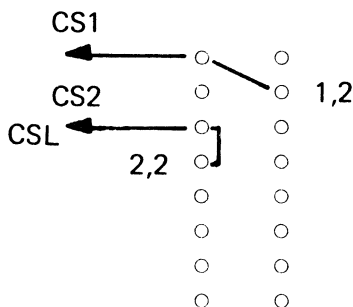
1 RAM 2kx8 (M58725P) in socket A2

We have to make the following connections on the board; First we define the start address of block A (8k) to 0000 by using a jumper from A to 0 on the platform.

0000 Hex	-----	
	1K RAM, internal	
0400	-----	
	1K RAM 4118	
07FF	-----	} 1 x 2K RAM M58725P
	2K RAM	
0FFF	-----	



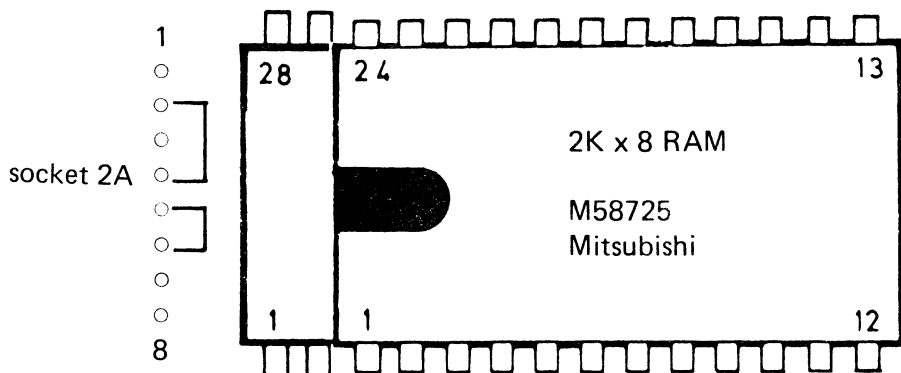
Now we have to define the start address of the sockets within block A. Socket 1A should start at address 0400 hex. For this we have to connect CS1 with 1,2. 1,2 selects addresses 0400-07FF. Socket 2A should start at address 0800 and should be selected for addresses up to 0FFF. To make this addressing possible we have to connect CS2 to 2,2. For details see chapter: THE 16k BYTE RAM/ROM BOARD.

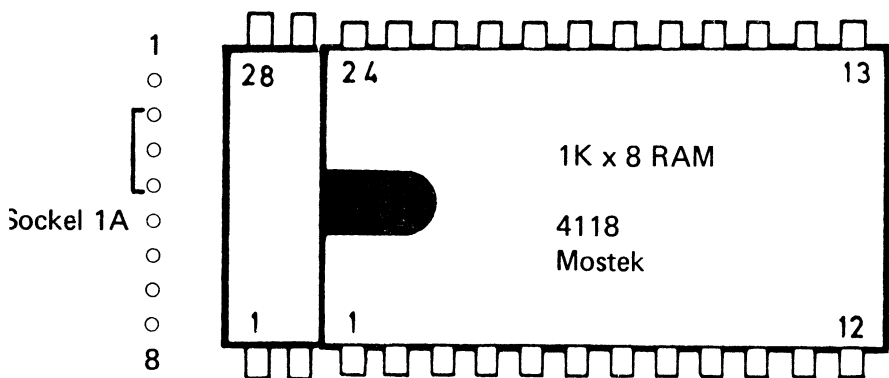


Our next step is to define the type of chip used in the two sockets. Socket 1A should hold a 1kx8 bit RAM. With this type of RAM we connect pin 2 and pin 4 at the line of pins beside socket 1A.

Socket 2A should hold a 2k RAM. For that we connect pin 2 with pin 4 and pin 5 with pin 6 at the line of pins beside socket 2A.

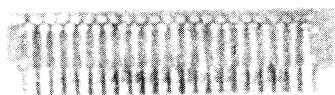
Now the card is prepared for the 3k RAM expansion. Next we need an adapter for the VIC-20 expansion port.



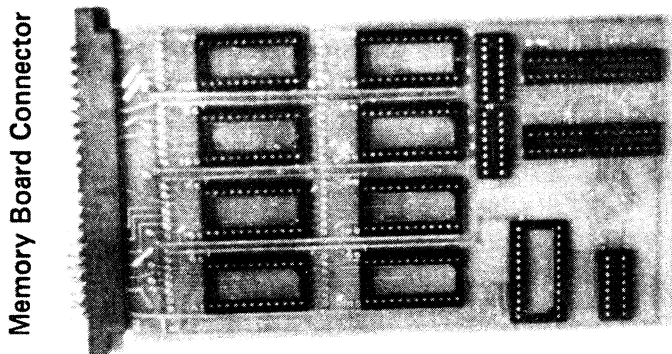


The adapter consists of a small board and a 44 pin plug. The board and the plug have to be connected according to the above table.

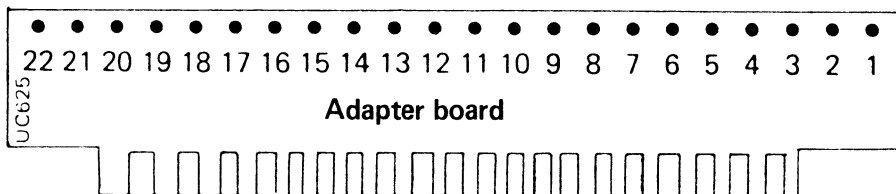
The addresses CA14 and CA15 are not at the expansion port of the VIC-20 and therefore have to be connected to two unconnected pins at the expansion port. These pins are Y and 20. To do this, we have to open the case (three screws).



Adapter Board



The populated RAM/ROM board and male connector for expansion connector



Now make the 30 connections using the following table:

VC-20 adapter board	Memory board connector	VC-20 adapter board	Memory board connector
1	1	A	A
22	22	B	6
2	H	C	5
3	7	D	F
4	8	E	4
5	J	F	D
6	T	H	E
7	S	J	W
8	15	K	V
9	16	L	19
18	12	M	17
20	Y	N	18
21	20	P	U
		R	C
		S	X
		Y	B
		Z	Z
		V	9

Connect: 1 → 1 A → A
 22 → 22 B → 6
 2 → H C → 5 a. s. o.

After you have made all the connections, check them one more time.

Block A (8k) of our board is exhausted now. The remaining portion is used internally in the VIC-20. Now we can define block B to start at \$2000 and fill with RAM up to \$3FFF (8k).

Another possibility is to use an EPROM starting at \$A000. If you have an EPROM starting at \$A000 you can have an autostart program by storing the start address of that program in \$A000 (lower byte) and \$A001 (higher byte).

Locations \$A004 to \$A008 have to contain the following:

```

$A004 : $41
$A005 : $30
$A006 : $C3
$A007 : $C2
$A008 : $CD

```

The first five commands in your autostart program have to be the following:

```

JSR $FD8D ; test of RAM
JSR $FD52 ; definition of vectors
JSR $FDFG ; I/O initialization
JSR $E518 ; initialization of screen
CLI      ; enable interrupt

```

Definition of start address \$A000 for block B:

We connect pin B on the board, and pin A to the right of the 74154 chip. Now the start address of socket 1B is \$A000.

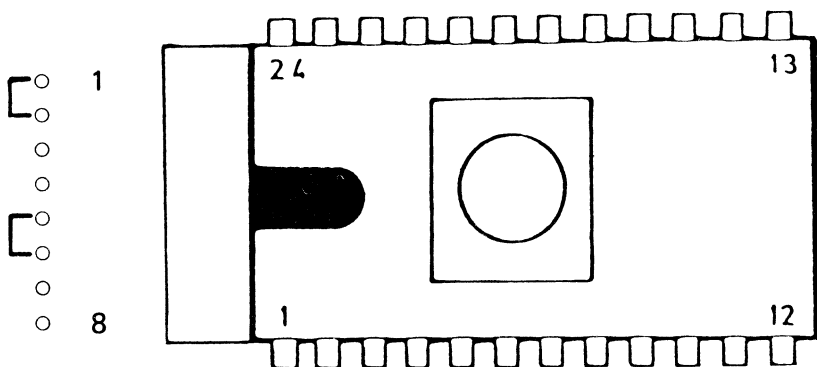
If we want to use four 2k EPROMs (2716) we have to make the following connections at the chip-select platform:



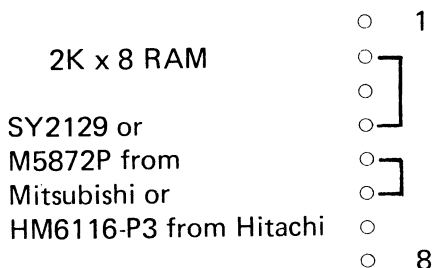
Decoding within the 8k block B

Now the start addresses of the four EPROMs are defined.

To be able to use 2716 EPROMs, we have to make two more connections at each of the four sockets:



Since the RAM/ROM board 615 is very versatile you also could use RAM-chips at \$A000, then the connections would look like the following:



If you want block B to start at \$2000 instead of \$A000, you just have to use a jumper from B to 2 rather than A.

16 K byte RAMROM-board

Memory expansion has never before been so interesting as today. Some of the new personal computers cry out for expansion from the moment they are first turned on. Our experience also shows that after satisfying an appetite for RAM, the computer enthusiast soon desires EPROM.

Consequently, ELCOMP has developed a memory expansion board with special features. The new memory expansion board (European size) allows the use of 16k RAM or 16k EPROM or a combination of RAM and EPROM. Our philosophy is founded on the byte-wide concept which permits the use of RAM chips compatible with 2716 EPROMs. For example, this means that one socket can hold a 2716 EPROM or a 4802 RAM from MOSTEK. Also possible is the use of 4K and 8K EPROMs (2732 and 2764).

This makes memory expansion very flexible because it allows on the same board, for example, a RAM expansion of 8k and an EPROM expansion with monitor, editor, assembler, etc. It is easy to change this configuration later to accommodate 17k EPROM or 16k RAM. The board has traces on both sides and uses the S44 standard bus. The 44 connections are marked by 1-22 and A-Z (similar to KIM, SYM, AIM, VC-20, ATARI, etc.). It is also possible to use it with our expansion board (ELCOMP-1, order # 606). Therefore, you can connect the board easily to almost any 6502 microcomputer.

You also can connect the board directly to your 6502 computer. Then you have to connect the microprocessor directly to the board.

The 16k RAM-ROM board is divided into two 8k blocks: A and B. You can define the start address of each block by decoding from addresses A 12 to A 15 on the IC 74154. Thus you can define the start addresses to \$0000, \$1000, \$F000. Within the 8k blocks the addresses are decoded by ICs 74LS138 and 74LS08 to steps of 1k and 2k with addresses A 10 to A 11.

Figure 1 shows the decoding of the RAM-ROM board. The decoding of the 8k blocks is the same for A and B. Figure 2 shows the distribution of the sockets among the blocks A and B as well as the numbering. The decoded signals for the 4k boundaries can be taken from a DIL socket at bottom right; the signals for 2k and 1k can be taken from 2 DIL sockets between the decoding and memory ICs. The upper socket is for block B, the lower for block A. The signals at these sockets are shown in Figure 3.

CS1-CS4 are the chip select signals for sockets 1–4 (pin 18 for 24-pin IC; pin 20 for 28-pin IC). 2,1–2,4 are the four 2k and 1,1–1,8 are the eight 1k decoded addresses.

Sample for block A:

Pin A is connected to the decoded addresses 2 and 3.

Pins 2,1 –2,4 decodes the following addresses:

2,1 =	\$2000	—	\$27FF
2,2 =	\$2800	—	\$2FFF
2,3 =	\$3000	—	\$37FF
2,4 =	\$3800	—	\$3FFF

Pins 1,1 – 1,8 decode the following addresses:

1,1 =	\$2000	—	\$23FF
1,2 =	\$2400	—	\$27FF
1,3 =	\$2800	—	\$2BFF
1,4 =	\$2C00	—	\$2FFF
1,5 =	\$3000	—	\$33FF
1,6 =	\$3400	—	\$37FF
1,7 =	\$3800	--	\$3BFF
1,8 =	\$3C00	--	\$3CFF

* 7408 Standard TTL
must be used
(no 74LS08)

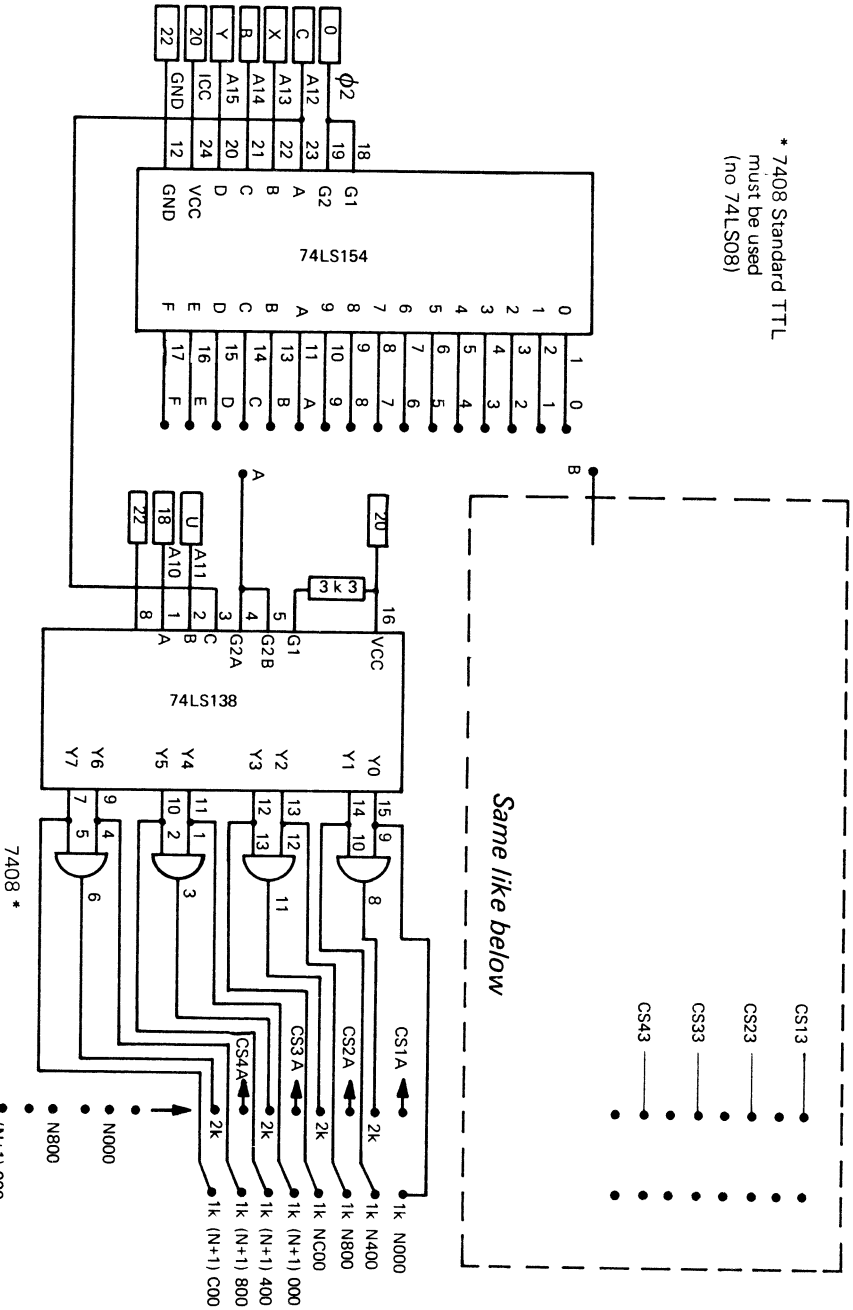


Fig. 1: Decoding on the RAM-ROM board

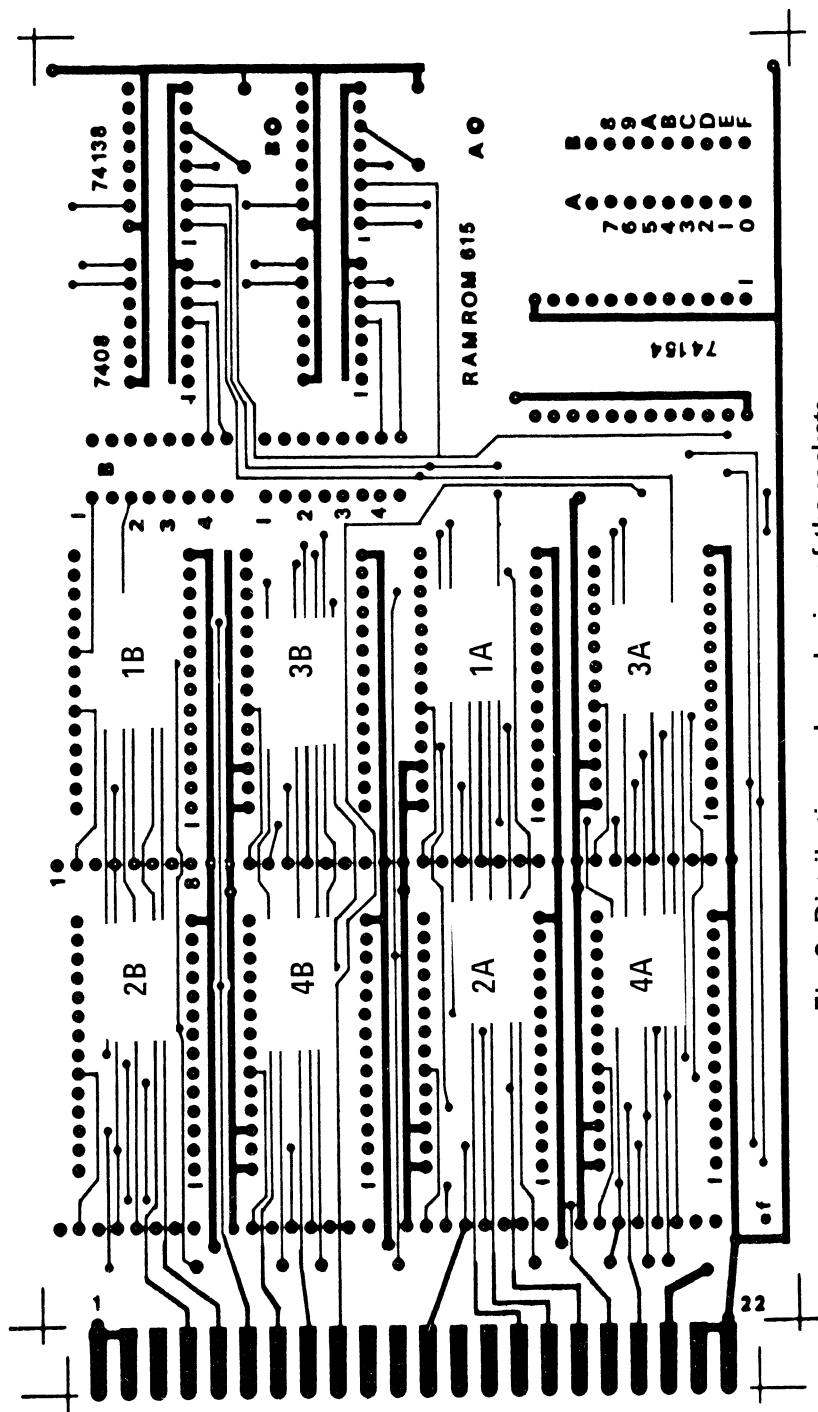


Fig. 2: Distributing and numbering of the sockets

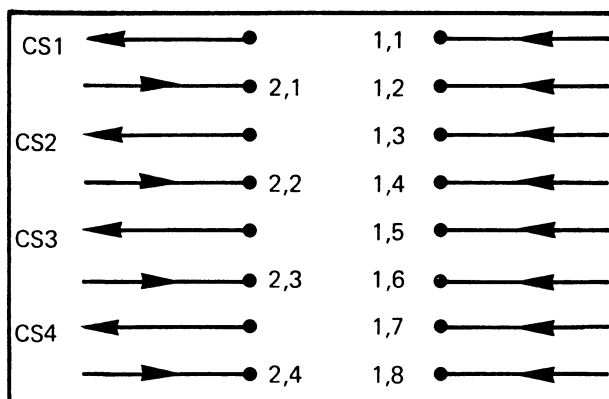


Fig. 3: Distributing of the decoded signals

If you put four 2k EPROMs or RAMs into the sockets, the following connections are needed: see Fig. 4a. The start address of socket A then is \$2000; the one of socket 2A = \$2800; socket 3A = \$3000; socket 4A = \$3800. If you use four 1k RAMs the connections are to be made according to Fig. 4b. Then the start addresses are as follows:

1A = \$2000
 2A = \$2400
 3A = \$2800
 4A = \$2C00

If you need the addresses \$3000 – \$3FFF, you must connect according to Fig. 4 c.

The old addresses at the sockets are at the lower pins (2,3 – 2,4 and 1,5 – 1,8); the even addresses are at the upper pins (2,1 – 2,2 and 1,1 – 1,4). This means if you connect A to pins 5 and 6 of the decoding socket you will get the following start addresses:

2,1 = \$6000	
2,2 = \$6800	
2,3 = \$5000	
2,4 = \$5800	and
1,1 = \$6000	1,5 = \$5000
1,2 = \$6400	1,6 = \$5400
1,3 = \$6800	1,7 = \$5800
1,4 = \$6C00	1,8 = \$5C00

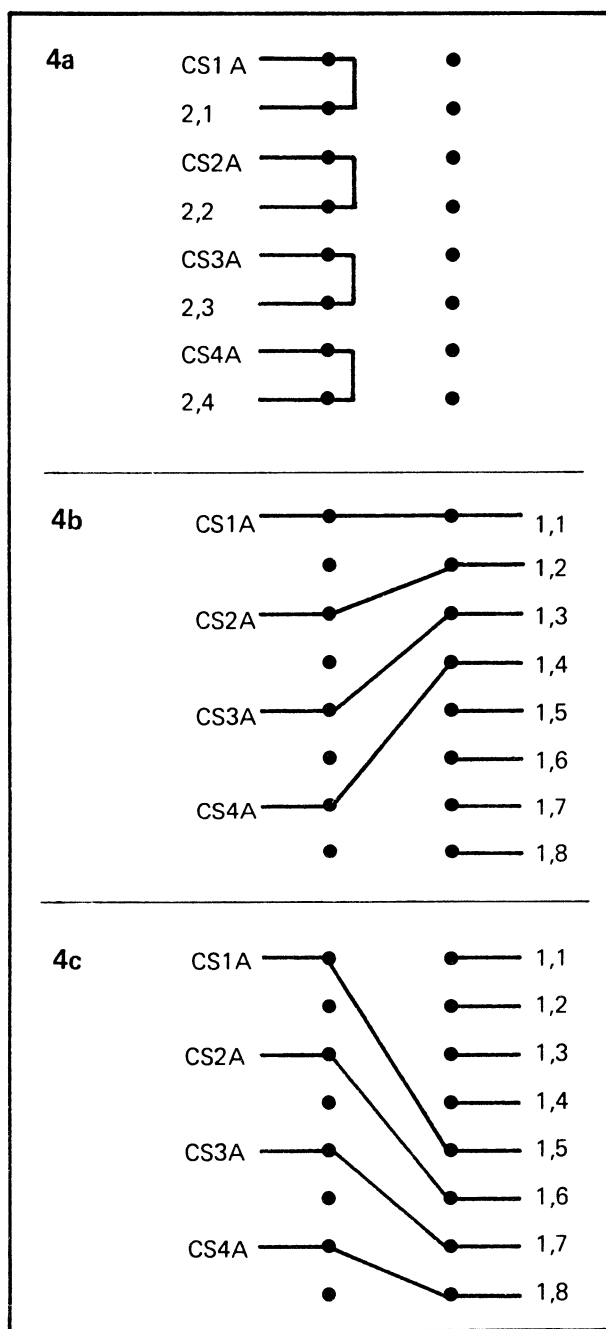


Fig. 4: Connections of the decodes signals to the chip select pins

Figures 5, 6, 7, 8 and 9 show different samples of connections.

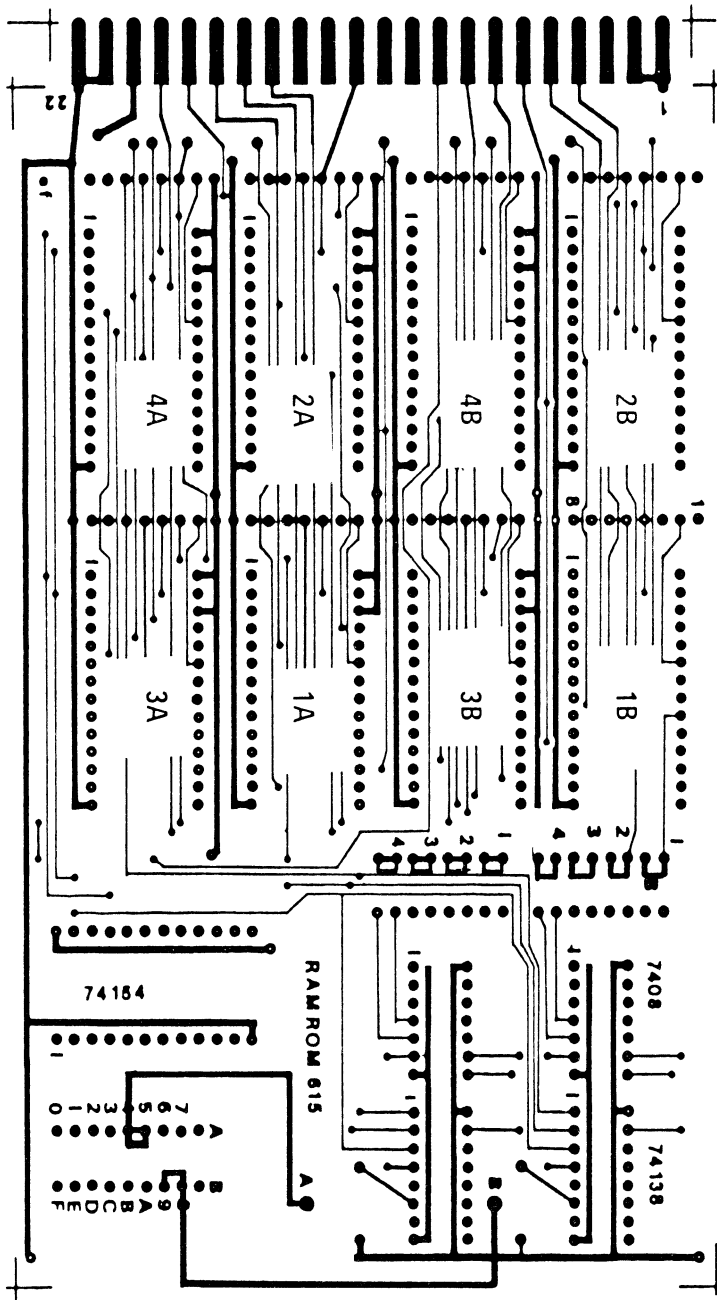


Fig. 5

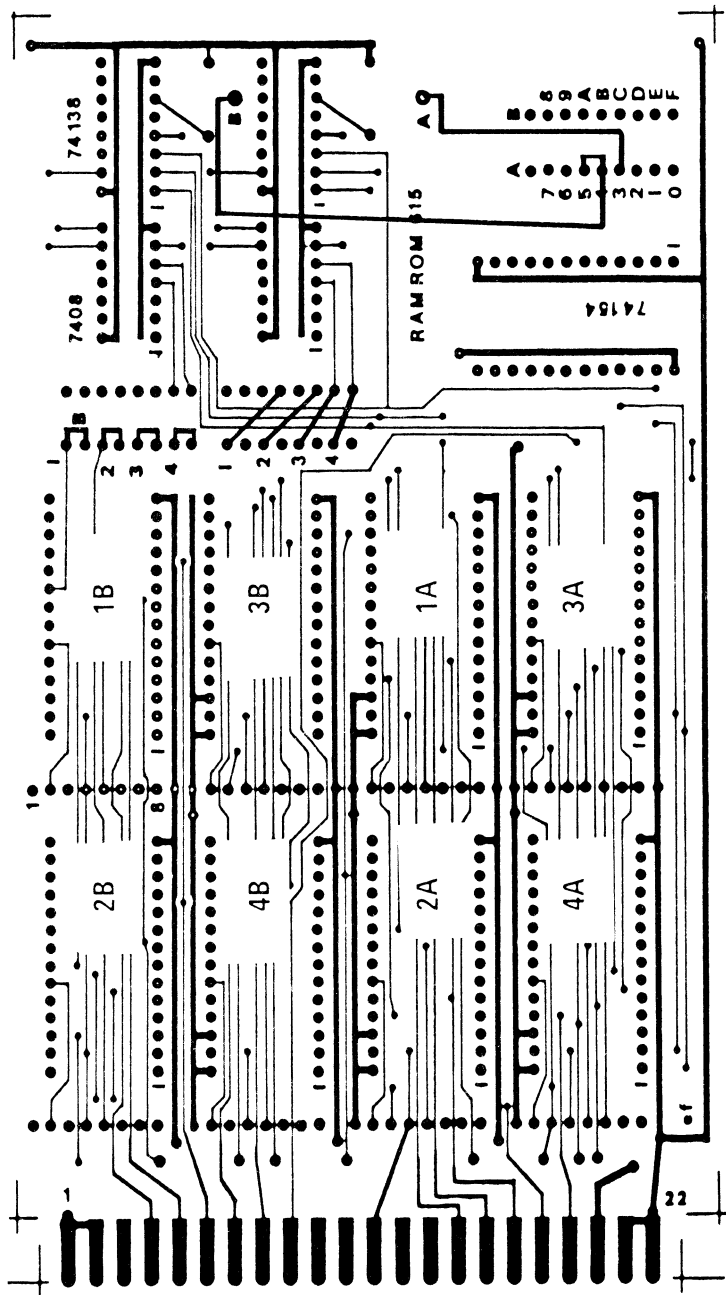


Fig. 6

Sample 1 (Fig. 5)

Sockets 1A — 4A,	4 x 2k RAM/EPROM	1A = \$4000	1B = \$8000
	Start address	\$ 4000	2A = \$4800 2B = \$8800

Sockets 1B — 4B	4 x 2k RAM/EPROM	3A = \$5000	3B = \$9000
	Start address	\$ 8000	4A = \$5800 4B = \$9800

Sample 2 (Fig. 6)

Sockets 1A — 4A,	4 x 1k RAM	1A = \$3000	1B = \$4000
	Start address	\$ 3000	2A = \$3400 2B = \$4800

Sockets 1B — 4B,	4 x 2k EPROM/RAM	3A = \$3000	3B = \$5000
	Start address	\$ 4000	4A = \$3800 4B = \$5800

Sample 3 (Fig. 7)

Sockets 1A — 4A,	4 x 2k EPROM/RAM	1A = \$4000	1B = \$6000
	Start address	\$ 3000	2A = \$4800 2B = \$6800

Sockets 1B — 4B,	4 x 2k EPROM/RAM	3A = \$3000	3B = \$5000
	Start address	\$ 5000	4A = \$3800 4B = \$5800

For example, if you use four 2k RAMs in an 8k block it does not matter whether the start address of ~~\$~~5000 is at socket 1A or at 3A. This is different from EPROMs. There the decoded address area must correspond with the contents of the EPROM so the program in the EPROM can run.

Here is another version of sample 3 (Fig. 8). In this sample the sockets are decoded in rising sequence:

1A = \$3000	1B = \$5000
2A = \$3800	2B = \$5800
3A = \$4000	3B = \$6000
4A = \$4800	4B = \$6800

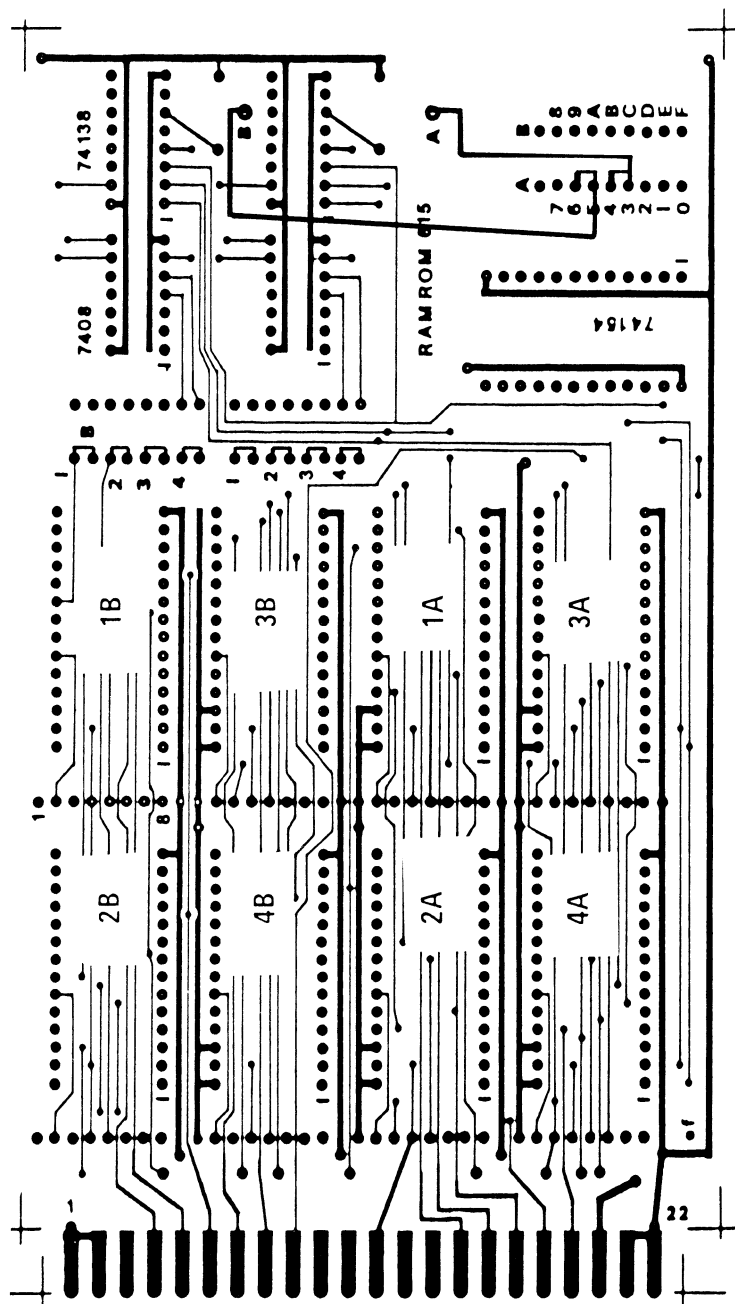


Fig. 7

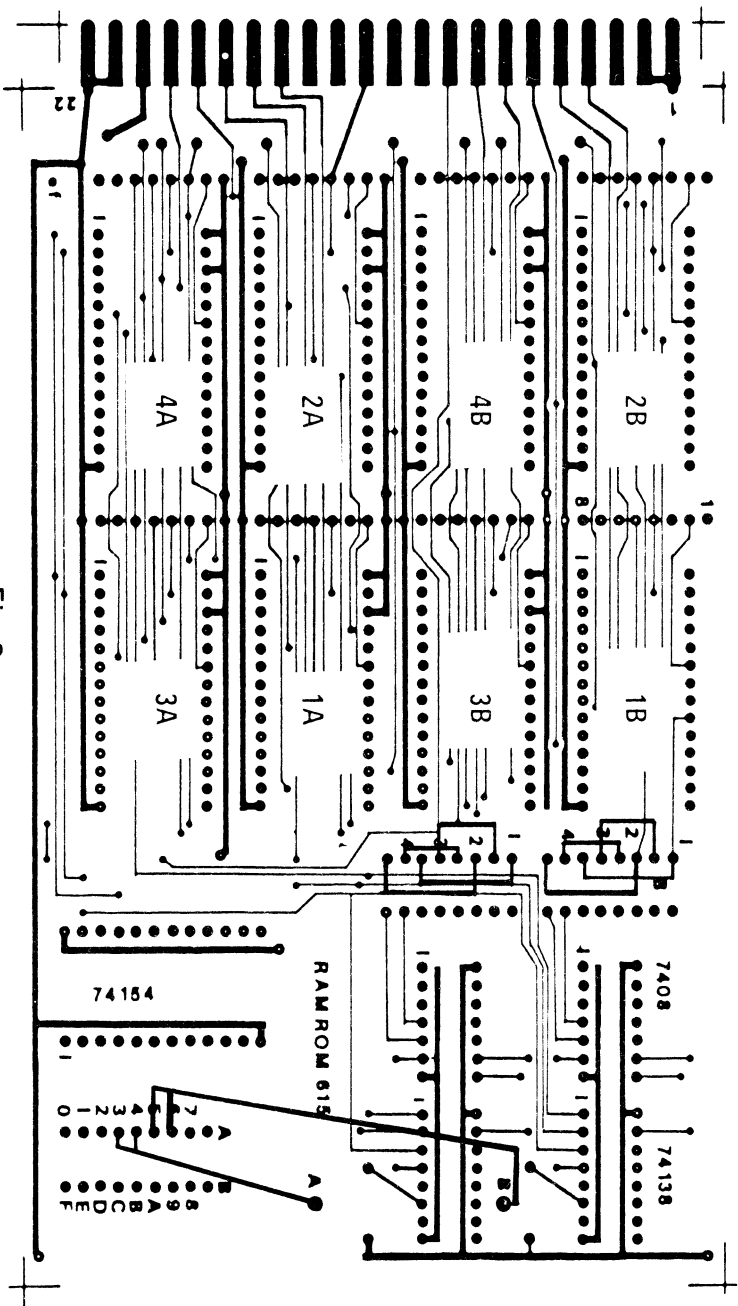


Fig. 8

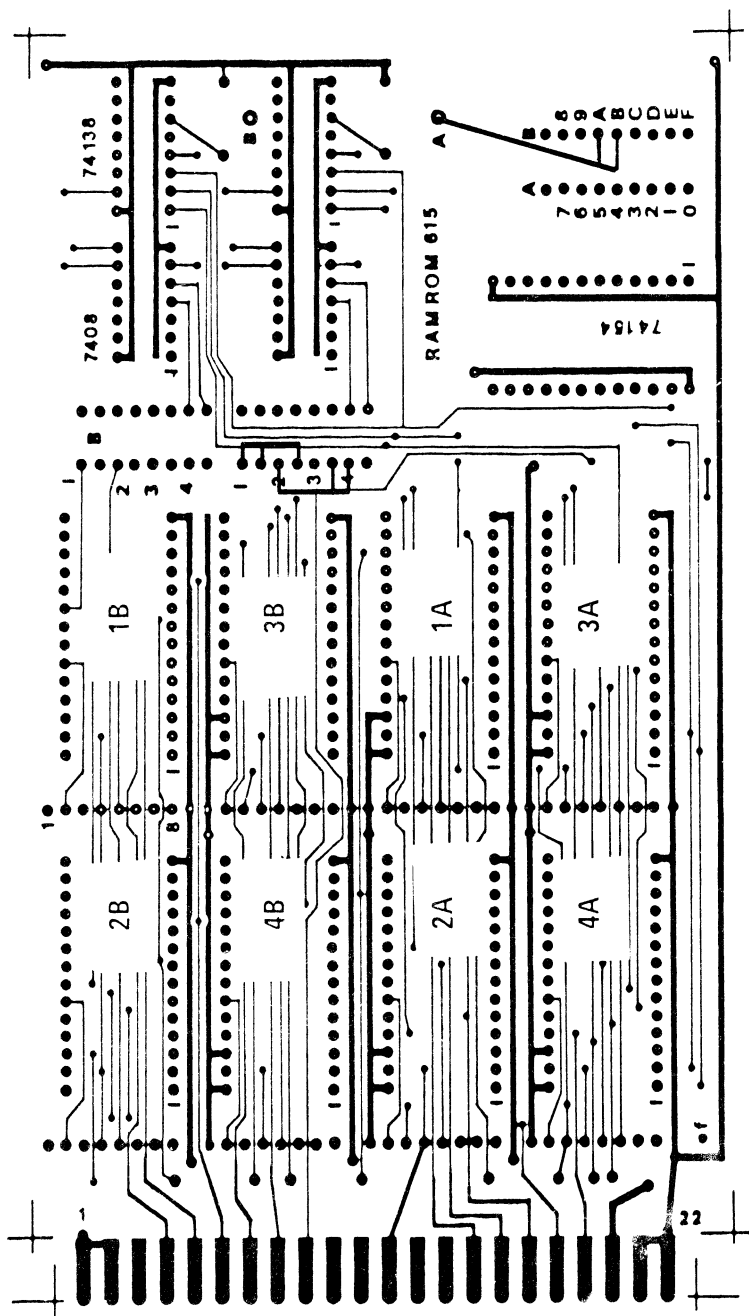


Fig. 9

Sockets 1A, 2A:

2 x 4k EPROM

1A = \$A000

Start addresses ~~\$A000~~, ~~\$B000~~

$$2A = \$3000$$

Next, definition must be made at each socket for what chip will be used (1k, 2kRAM or 2k, 4k EPROM). To do that, there is a line of pins at each socket. Fig. 10 shows a socket and this line. The signals A11, WE, A10 come to the line from outside and there are connections to the socket: pin 19 (21) and 21 (23). The numbers in parentheses are for the 28-pin ICs. Fig. 11 shows the connections for 1k x 8, 2k x 8RAM and for 2716, 2732 EPROMs. A11 OE lines are connected to GND.

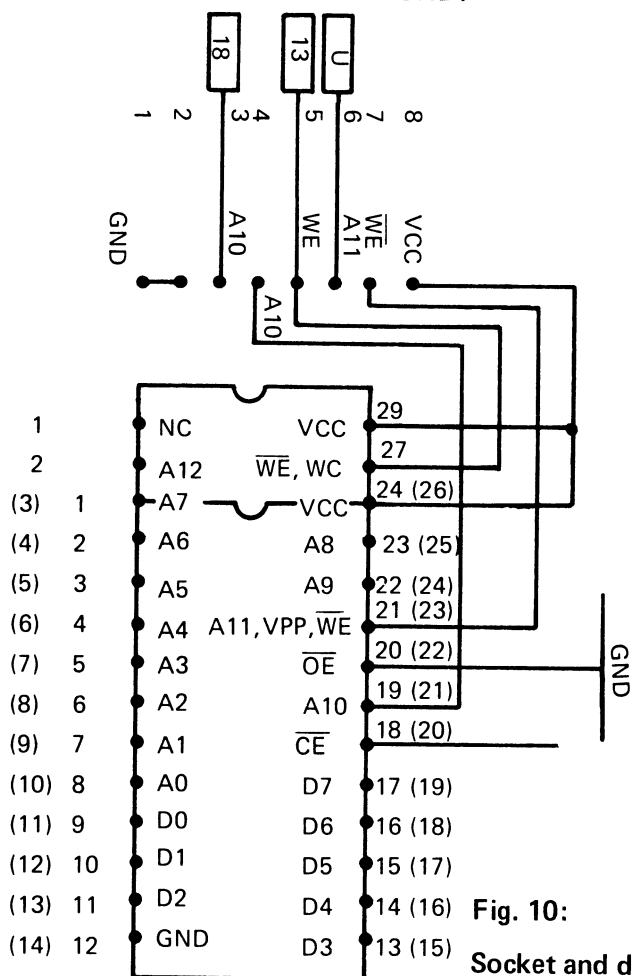


Fig. 10:
Socket and distributing line

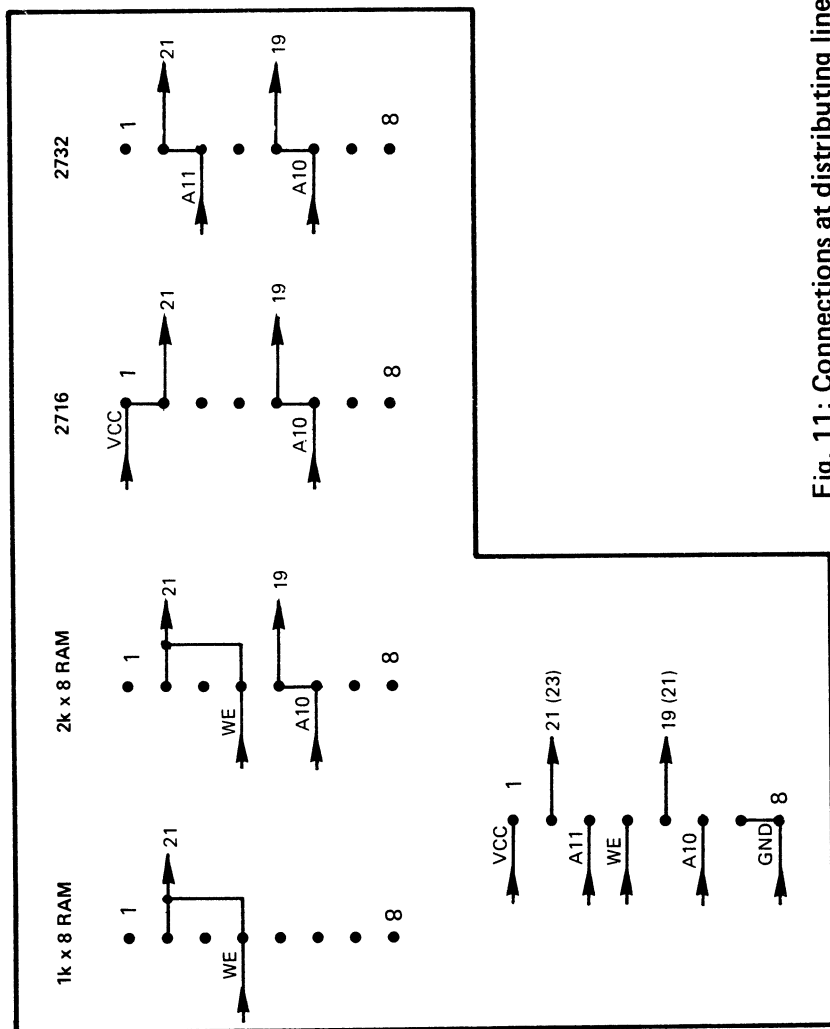


Fig. 11: Connections at distributing line for different ICs

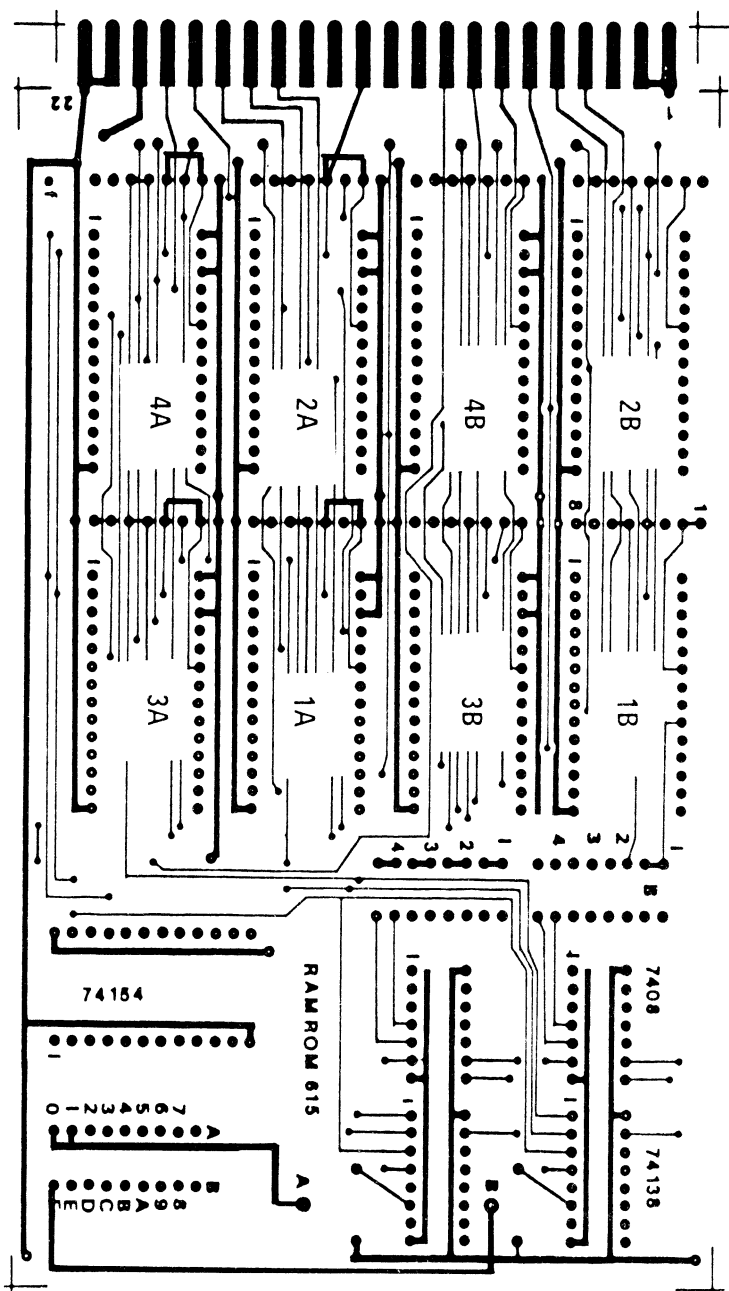


Fig. 12

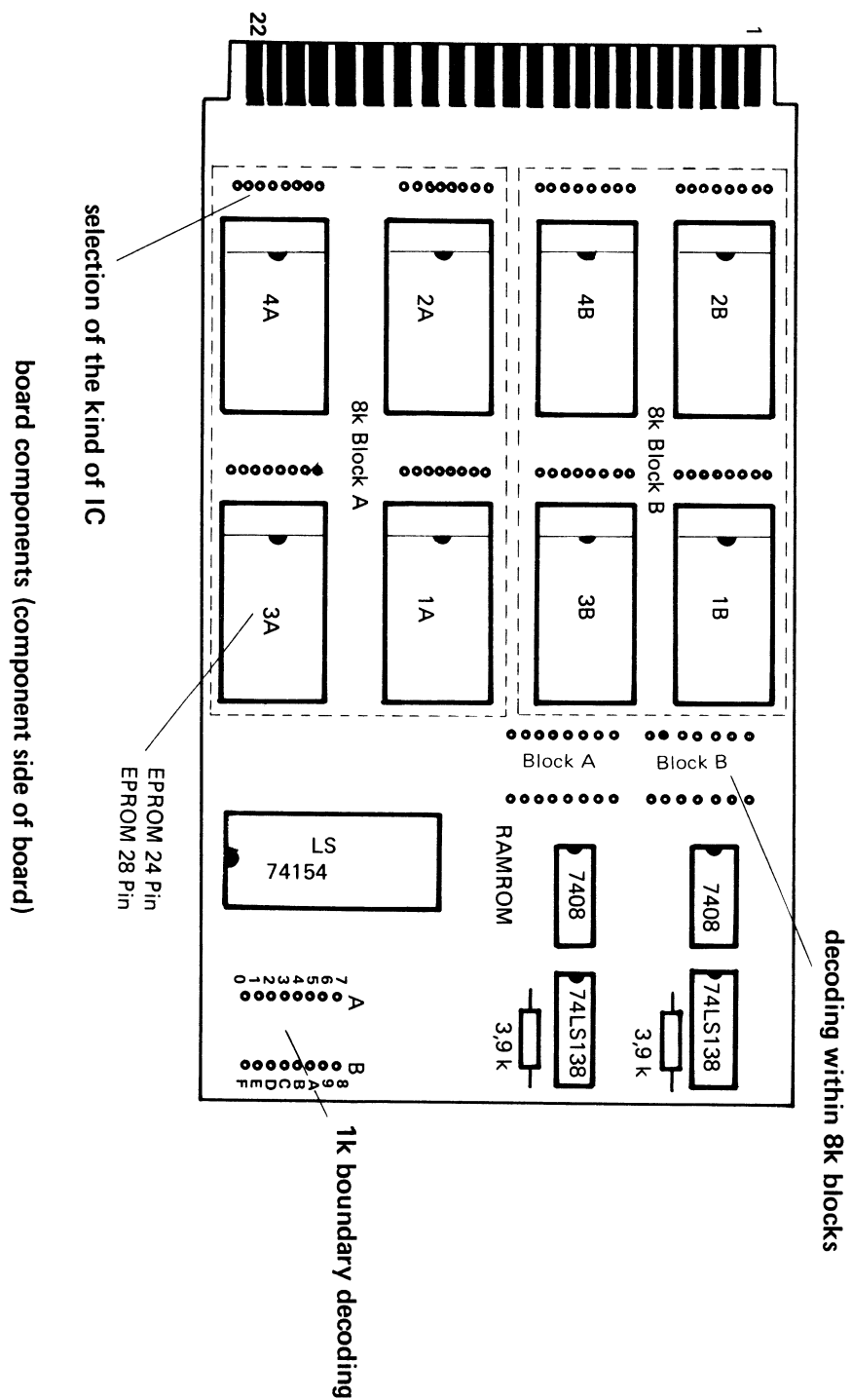
The last sample shows a memory expansion with 8k RAM from \$0000 to \$3FFF and 2k EPROM from \$F800 to \$FFFF. The sockets 1A — 4A contain RAM; socket 1B contains a 2716 (Fig. 12).

Fig. 12 socket 1A — 4A, 4 x 2k RAM
 socket 1b , 1 x 2k EPROM 2716

The last figure shows the signals at the bus, similar to the S44 bus. The same configuration of signals is on the ELCOMP-1 board.

	S-44E		S-44E
1	GND	A	GND
2	GND	B	A14
3	NC	C	A12
4	A3	D	A4
5	A1	E	A5
6	A0	F	A2
7	D1	H	D0
8	<u>D2</u>	J	D3
9	$\phi 2$	K	NC
10	NC	L	NC
11	NC	M	NC
12	R/W	N	NC
13	NC	P	NC
14	NC	R	NC
15	D6	S	D5
16	D7	T	D4
17	A9	U	A11
18	A10	V	A7
19	A8	W	A6
20	VCC	X	A13
21	GND	Y	A15
22	GND	Z	GND

Armament of the board: The following figure shows the 16k RAM-ROM board with the positions of the ICs. For the use of EPROMs are EPROM-compatible RAMs, 28-pin sockets are designated.



If you use ICs 2716 or other 24-pin ICs, four pins will remain empty on the left side of the socket. To the left of each socket you can see eight soldering dots where you can put a strip which may be wired with jumpers, matching the IC used. For decoding the 8k blocks there are two rows, each with 16 soldering dots where you can connect two 16-pin DIL sockets. The best way is to plug 16-pin DIL platforms (adapters) where you can easily connect the jumpers needed for the decoding. At bottom right there is the decoding for the 1k boundaries. The best way in this case also is to connect your jumpers on a platform.

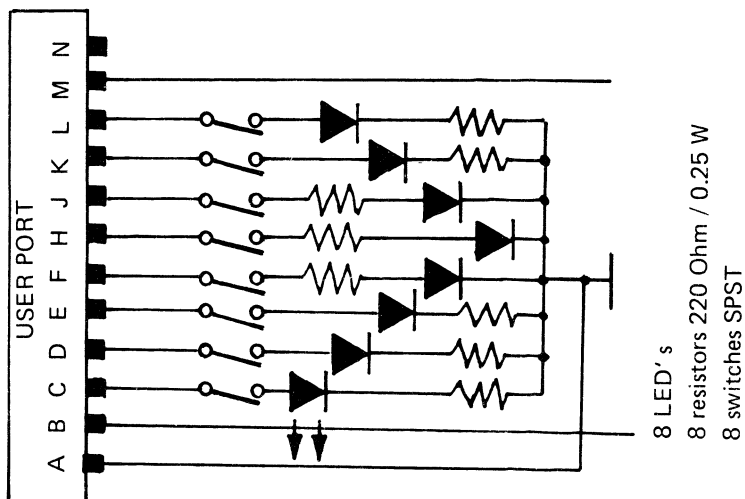
Input/Output programming

INPUT OUTPUT PROGRAMMING WITH YOUR VIC-20

The VIC-user port with its 6522 VIA chip allows the experimenter to do his own input/output programming. This means that you can send data from the computer into an external device, such as an alarm system, or a power switch for your TV, radio, stereo, or coffee pot, etc. You can use your computer as a powerful programmable timer, or control unit. You also may input data from external devices and bring information (binary) from external sources into your computer. Now you can use this data, do the specific calculations and then store it in RAM or output it again into an external device.

Here are some practical examples:

You have eight switches which are connected with the eight bits of the VIA input port.



These switches are connected with your light switches in each room of your home or apartment. If one of the switches is closed you will have 0 volts on a specific input. Now you can write a computer-program, which checks all the switches whether they're closed or open. This gives you the direct information as to whether the lights in a particular room are switched on or off. Your computer program also can measure the time a switch was open or closed during a twenty-four hour period. All this information can be stored and recalled after a certain time interval. I.e. after two weeks the computer can give you a report, how long each switch was closed or open over that period. You even can use the information and calculate the money you spent for electricity during that period. This example is only one of thousands of things you can do with an inexpensive homecomputer, and a little knowledge of input/output programming.

Recommended parts

For your experiments we recommend the following parts:

- 1.) 1 power supply DC + 5V/1A
- 2.) 1 prototyping board similar to the "global specialties" protoboard PB 101 from ACP in Santa Ana, CA.
- 3.) 1 15 conductor ribbon cable approx. 2 feet long.
- 4.) 1 connector for the VIC-20 user port (24 pin) from TRW Circle 251-12-90-160 type 5024ee-30) from Advanced Computer Products in Santa Ana, CA
- 5.) 3-8 switches (on/off) (spst).
- 6.) 3-8 leds and 3-8 220 /0.25W resistors.

Bring out the eight output ports of the VIC-20 (PB0-PB7) via the ribbon cable to the prototyping board PB101. Now you can connect the switches and the LED's to the ports.

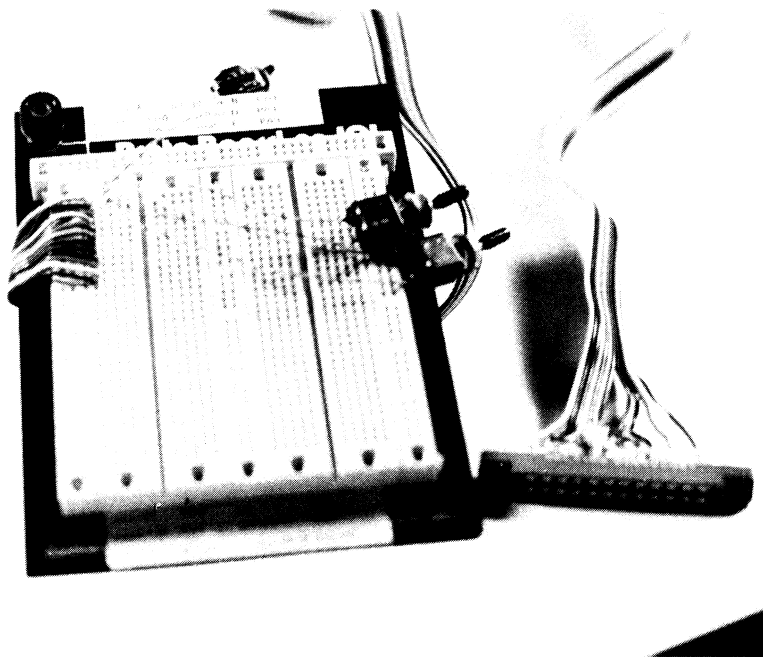


Figure 1 shows you a sample with the port lines of port B connected to a LED circuitry on a proto board. Lines PB0-PB7 are connected.

THE 6522 VERSATILE INTERFACE ADAPTER VIA IN THE VIC-20

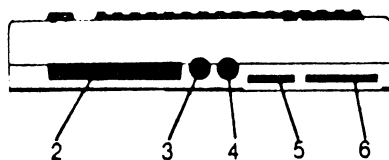
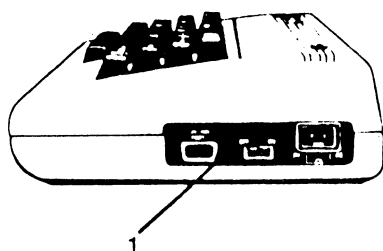
The 6522 VIA, which is the heart of the VIC-20's user port, is an extremely powerful I/O chip for 6502 based microcomputer systems. The 6522 VIA versatile interface adapter constitutes a significant improvement over the original 6820 PIA. The 6522 adds two internal timers, a serial in/out and parallel in, serial out shift register, and latched data input on the peripheral ports. Enhanced handshaking permits high-speed data transfer between multiple CPUs.

The primary feature of the 6522 VIA is a pair of 8-bit bidirectional data ports. Each of the 16 bidirectional lines may be programmed as either an

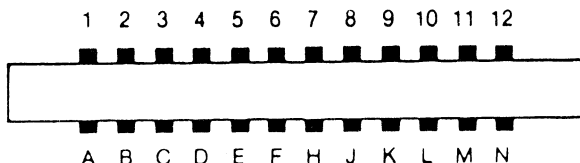
input or output. Some of the lines may be interfaced with the internal interval timers providing the ability to generate specific time intervals, or to measure specific time intervals. The 6522 is controlled by an internal file of 16 8-bit registers.

A SAMPLE SESSION WITH THE PROTOTYPING BOARD

For your input/output programming experiments, we recommend to hook up the protoboard as described above. Plug in the connector into the user port #6.

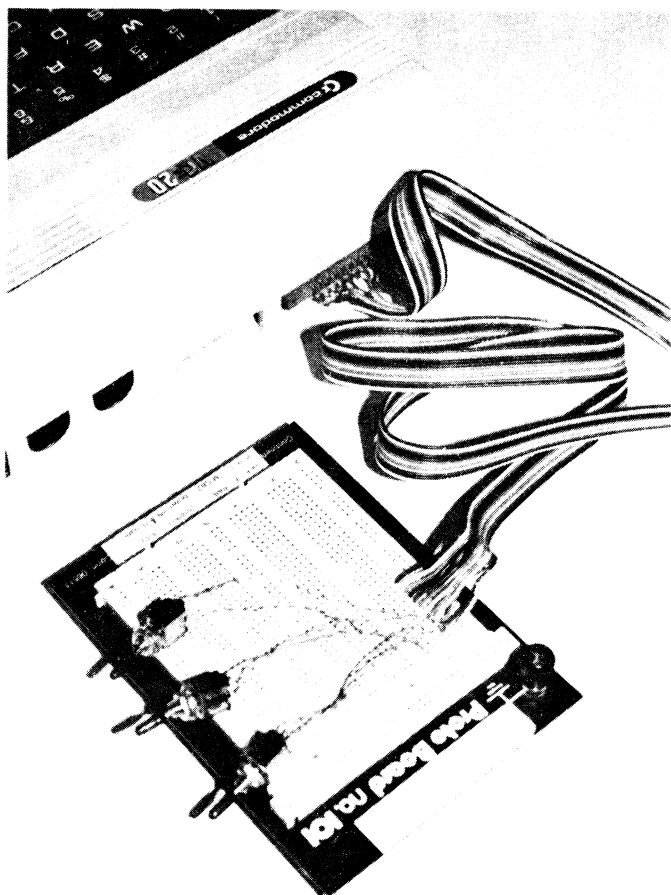


USER I/O



- 1 = joystick
- 2 = expansion
- 3 = audio/video
- 4 = serial port
- 5 = cassette
- 6 = user I/O port

PIN #	TYPE	NOTE	PIN #	TYPE	NOTE
1	GND	100mA MAX.	A	GND	
2	+5V		B	CB1	
3	RESET		C	PB0	
4	JOY0		D	PB1	
5	JOY1	100mA MAX.	E	PB2	
6	JOY2		F	PB3	
7	LIGHT PEN		H	PB4	
8	CASSETTE SWITCH		J	PB5	
9	SERIAL ATN IN		K	PB6	
10	+9V		L	PB7	
11	GND		M	CB2	
12	GND		N	GND	



Note: Verify your VIC-20 is turned off prior to attachment of lines.

It is also very useful if you put a label on your protoboard indicating corresponding port lines to wire color.

red=PB0	blue=PB2	white=PB5	green=PB7
black=PB3	pink=PB6	yellow=PB4	orange=PB1

When everything is hooked up properly, you can switch on your VIC-20. You are now in a position to

send and receive data with your prototyping board. An 8 bit pattern can now be sent to external circuitries and devices.

For your first experiment we will give you a small program which allows you to look at either input or output data in binary form on the prototyping board ports. Type in this program and run it. The computer first asks you whether you want to input data or output data via the I/O port. If you type in "1" for inputting data, the display shows you whether or not anything is connected. 255=decimal and 11111111=binary as the momentary status of the input port. This means that all switches are open and all inputs are logical "1". You can test that while you change the input information with the switches. Close one or more switches and watch the numbers on the screen. If you are unfamiliar with number systems and the conversion between different bases, you should at this time refer to a base conversion chart. Almost any primer book contains the basics of number systems.

```
5 REM COPYRIGHT BY
7 REM WINFRIED HOFACKER
10 PRINT"THIS PROGRAM"
12 PRINT"SHOWS THE STATUS"
14 PRINT"OF THE USER PORT IN"
16 PRINT"DECIMAL UND BINARY"
20 PRINT"J"
39 S$(0)="0":S$(2)="1"
40 GOTO 200
41 POKE 37138,0
60 PRINT"DECIMAL ";PEEK(37136)
61 PRINT:PRINT"BINARY"
62 P=PEEK(37136)
63 GOTO 90
65 FOR I=1 TO 8
70 PRINT P AND 1:P=P/2
80 NEXT I
81 PRINT"BINARY"
90 PRINT"||"ABS((PAND128)=128);
91 PRINTABS((PAND64)=64);
```

```

92 PRINTABS((PAND32)=32);
93 PRINTABS((PAND16)=16);
94 PRINTABS((PAND8)=8);
95 PRINTABS((PAND4)=4);
96 PRINTABS((PAND2)=2);
97 PRINTABS((PAND1)=1);
98 FOR V=1TO 3000:NEXT
110 PRINT"CONTENTS OF DATA DIRECTION"
111 PRINT"REGISTER DDR";PEEK(37138)
112 FOR X=1 TO 3000:NEXT
120 GOTO 60
200 PRINT"INPUT='1',OUTPUT='2'"
210 INPUT N
220 IF N=1 THEN 41
230 IF N=2 THEN 300
300 REM OUTPUT OF INFORMATIONS
305 POKE 37138,255
310 PRINT"WHAT BITS"
330 INPUT K
340 IF K>255 THEN 310:GOTO 300
350 POKE 37136,K
370 L=PEEK(37136)
380 GOTO 60

```

If you type in "2", you can output information to the port and to your prototyping board.

A logic probe, which you can obtain from almost any electronic-parts store, would be a great help in identifying all the ports. You don't need it if you use your own LED indicators as shown previously.

It is strongly recommended to experiment for a while with the program, switches, and LED's.

Close a few switches and use function 1 of the program to check if the binary representation exactly corresponds to the numbers on the screen.

Input/output programming with the VIC-20

The VIC-20 has, as mentioned before, the 6522 VIA chip wired to the user I/O port. Unfortunately there

are only 8 I/O lines, PB0-PB7, available for the user. The remaining lines, PA0-PA7, are used for the joystick, cassette switch, light pen, and some other internal features of the VIC-20.

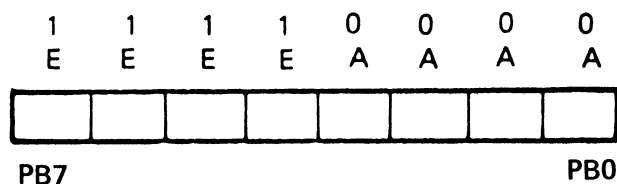
Each of those I/O lines (PB0-PB7) can be used as an input or an output dataline. To set the direction of the dataflow you just set the data-direction registers with:

POKE 37138,0 all ports are input

POKE 37138,255 all ports are output.

If you only want four of the eight data lines as inputs and the rest to remain output lines, you have to enter :

POKE 37138,240



Now the lines PB0-PB3 are input and lines PB4-PB7 are output. 240 is the decimal equivalent of the number 11110000.

Thus you can set any combination of input/output lines between 0 and 255. With this poke statement you now have set the direction of the data. Now you can send data out or check incoming data from the outside of the computer.

After power up, the VIC-20 sets all I/O lines as inputs. You can check this while you type in the following:

10 ? PEEK (37138)
RUN

The VIC should print out a 0.

Now we can try to output data to the port. The VIC (resp. the 6522) has a certain address where you can put the data in.

Lines PB0-PB7 37136 decimal

Everything you poke into this location appears now at the output port.

With PEEK 37136 you can read the information which is brought to the port lines. The DDR (data direction register) has to be set to input or output prior to this step. If the port lines are open (not connected to +5V or 0V) the PEEK 37136 command will supply a logical one.

The following minidemo allows you to set the datalines and look at them afterwards.

```
10 REM MINIDEMO
20 POKE 37136,255
30 FOR I=0 TO 255
40 POKE 37136,I
50 FOR J=0 TO 255
65 FOR Y=1 TO 1000:NEXT Y
70 PRINT"THE CONTENTS OF"
72 PRINT"THE I/O"
73 PRINT"REGISTER IS: ";PEEK(37136)
80 NEXT I:NEXT J
```

The program above sets the output ports PB0-PB7 one after the other to each possible combination. After the loop is finished the output remains 255=11111111.

You now can modify this program to output a bargraph - like combination. This means that one line after the other is switched on and the previous line is switched off. To check this you can connect LEDs and resistors to each portline (PB0-PB7) and can watch the blinking lights.

```
100 REM MINIDEMO 2
110 REM RUNNING LIGHT
120 REM THE NUMBER 900
130 REM IN LINE 165
140 REM DEFINES THE
141 REM SPEED
```

```

142 POKE 37138,255
143 FOR I=1 TO 7
144 X=2↑I
145 PRINT X
146 POKE 37136,X
147 FOR J=0 TO 255
165 FOR Y=1 TO 900:NEXT Y
170 PRINT"THE CONTENTS OF"
172 PRINT"I/O-REGISTER"
173 PRINT"IS: ";PEEK(37136)
180 NEXT I
195 GOTO 143

```

The data can also be obtained from DATA statements.
(see the following program)

```

200 REM MINIDEMO3
210 READ NUM
220 IF NUM>128 THEN RESTORE:GOTO210
225 FOR G=1 TO 900:NEXT G
230 POKE 37136,NUM
235 PRINT PEEK(37136)
240 GOTO 210
300 DATA 1,2,4,8,16,32,64,128,255
310 END

```

For study of the internal life of the 6522 chip and for better understanding of this powerful chip, we recommend to take a look at the R6500 Hardware Manual from Rockwell or the Third Book of Ohio Scientific from us.

IMPORTANT NOTICE:

If a port or all ports are set as outputs, do not connect a port to + 5V or to ground without a 330 Ω resistor in series. When set as outputs you can connect the ports either to + 5V or to ground. Never use a voltage greater than 5V DC.

Input/Output

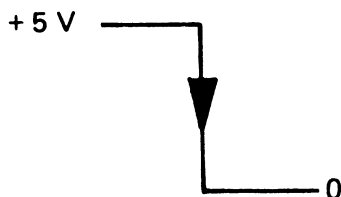
programming –

Part II

INPUT/OUTPUT PROGRAMMING – PART II

Until now we only have dealt with eight port lines of the VIC-20 user port. Port B of the 6522 VIA chip however, also has very powerful handshaking and interrupt capabilities. For this purpose, pins CB1 and CB2 can be used to bring in information from the outside. This may happen only when a specified pulse occurs on CB1 or CB2.

After powering up, a negative transition on CB1 allows the VIC-20 to bring in information from the outside via the handshake port.

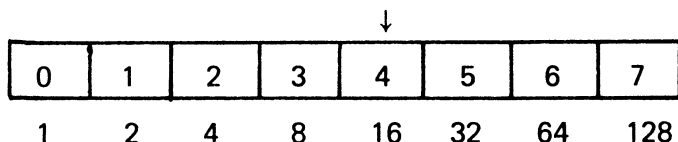


If the peripheral device or your outside circuitry generates a positive transition from 0V to +5V you also can use this as a trigger. Therefore, you have changed the peripheral control register inside the VIA chip. Bit 4 of this register sets the condition for a positive or a negative transition on CB1. It is originally set to zero and can be set to one by the following POKE command.

```
POKE 37148,PEEK(37148)OR16
```

Now, after a positive going pulse appears on CBI (Pin B of the VIC-20 user port) the interrupt flag will be set and the computer knows that an external device is ready to send its information via the 8 I/O lines.

Bit 4 in the Interrupt Flag Register (IFR) is affected.



THE INTERRUPT FLAG REGISTER (IFR)

The interrupt flag register has the decimal address 37149. From BASIC you can access this register with the WAIT 37149,16 statement.

The logical AND leads to log."1" only if both values are log."1". This means that the content of the flag register (37149) has to be ANDed with 0001 0000 (16) and checked if it's logical "1".

Normally bit 4 of the control register is "0". If the statement WAIT 37149,16 is executed, the logical AND-function result is zero. This means that BASIC will not work until a positive going pulse appears which makes the control register a "1". Please be careful with this statement and don't use it unless you can supply a +5V to CBI on the user port.

```
10 A$="  "  
100 A$="  "  
120 FOR J=1 TO 72  
130 WAIT 37149,16  
140 N=PEEK(37136)AND127  
150 IF N=13 THEN 180  
160 A$=A$+CHR$(N)  
170 NEXT J  
180 PRINT A$
```

The sample above allows you to verify what we have learned so far.

After executing the WAIT statement, data can be read from the user port PB0-PB7 into the computer.

N = PEEK (37136)

The read operation N = PEEK (37136) resets CB1 so that it can be checked for another set of data. 37136 is the address for the input/output register B containing handshaking. The content of the register is brought in and assigned to the variable N.
(see program listing above)

The value for N in this case could be between 0 and 255. It depends on how the switches are set on the prototype board.

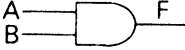







The following program allows you to read in ASCII Code.

```
10 DDR=37138
20 ACR=37147
30 FR=37149
40 PCR=37148
100 POKE DDR,0
110 POKE ACR,28
130 POKE PCR,16
135 PRINT"OPERATE"
140 WAIT FR,16
142 N=PEEK(37136)
145 PRINT PEEK(37136)
150 PRINT"AAA"
```

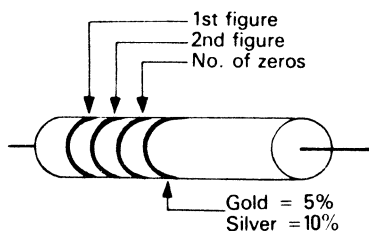
You can read in up to 72 ASCII characters. The input is limited to 7 bits.

```
10 REM MINIDEMO CB1 INTERRUPT
20 PRINT"HOFACKER"
30 PRINT"USE CB1"
40 PRINT"APPLY +5V"
50 WAIT 37149,16
60 PRINT" AAA"
```

Logic symbols

AND gate		$F = A \cdot B$
NAND gate (negated output)		$F = \overline{A \cdot B}$
NAND gate (negated inputs)		or $F = \overline{A} \cdot \overline{B}$
OR gate		$F = A + B$
NOR gate (negated output)		$F = \overline{A + B}$
NOR gate (negated inputs)		or $F = \overline{A} \cdot \overline{B}$
EXCLUSIVE – OR gate		$F = A \cdot \overline{B} + \overline{A} \cdot B$
Inverter gate.		$F = \overline{A}$

Color code for resistors



1,000 Ohms = 1K Ω
1,000,000 Ohms = 1M Ω

0 Black	5 Green
1 Brown	6 Blue
2 Red	7 Violet
3 Orange	8 Grey
4 Yellow	9 White

Code for capacitors

Colour Codes:

- as for resistors
- first three bands
- units are pF (pico-Farads)

Number of Markings:

- mF : micro Farads.
- nF : nano Farads.
- pF : pico-Farads
- Π : (Russian 'P')
- pico Farads
- H : (Russian 'N')
- nano Farads

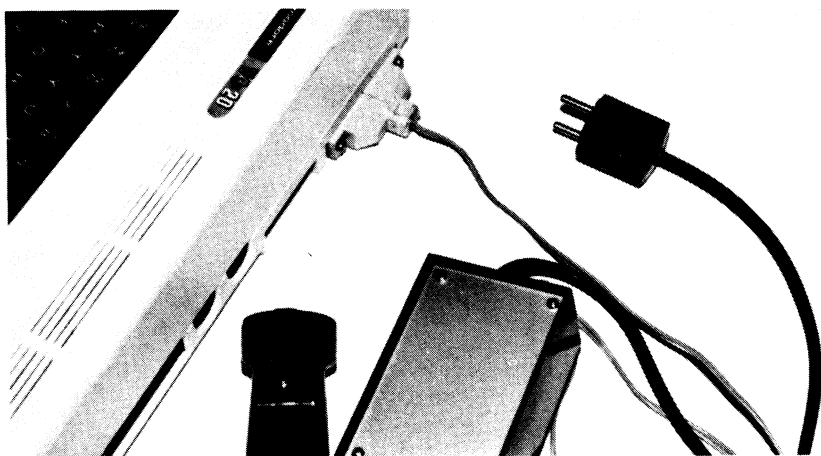
Numerals only:

- first two digits are the value.
- Third digit is number of following zero's.

Interface for power system users

INTERFACE FOR POWER SYSTEM USERS

For those of you who want to use the VIC-20 for control applications we have developed an interface that allows you to turn power systems on and off. The interface can be plugged in at the USER I/O connector on the back side of the VIC-20.



There are two relays in that interface so that you can control up to two independent devices. Port B of the VIA 6522 is connected to the USER I/O connector. PB0 - PB7, CB1 and CB2 are connected as well as 5V and GND. Port A of that chip is used for the joystick, the lightpen, and control of the tape recorder.

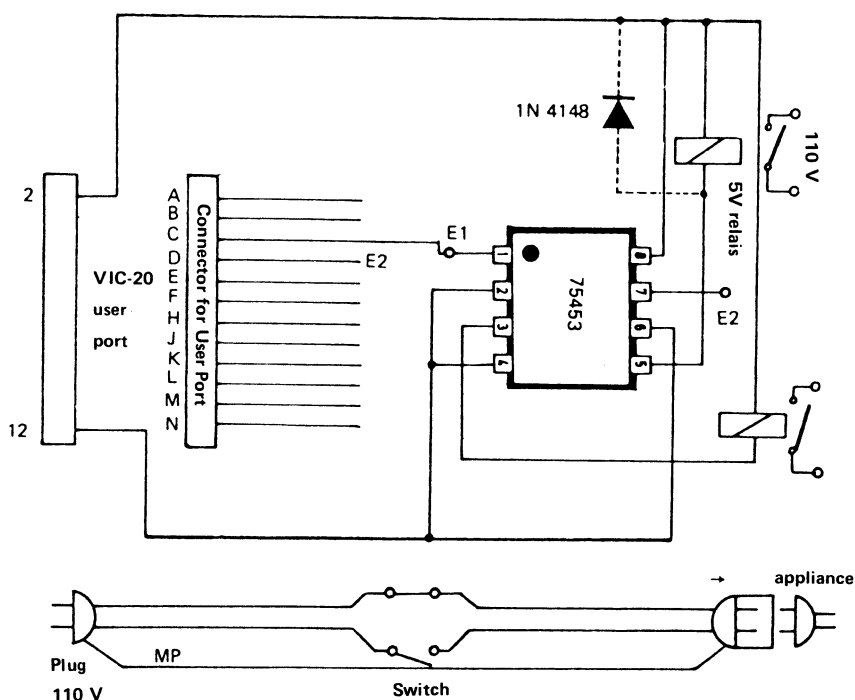
Caution: The maximum permissible load of the port is 100mA.

We use a TRW CINCH plug (TRW 251-12-160/50-24A-30) to connect our interface to the user I/O port.

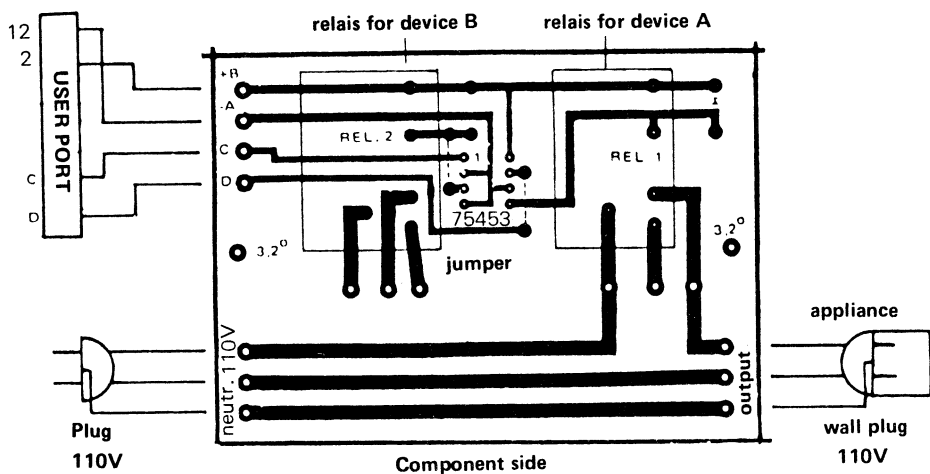
E1=input of relay 1 at PB0=C

E2=input of relay 2 at PB1=D

The input of relay 1 (E1) is to be connected to PB0 (at pin C). E2 can be connected to PB1 (at pin D). In the following circuit only one relay is connected.

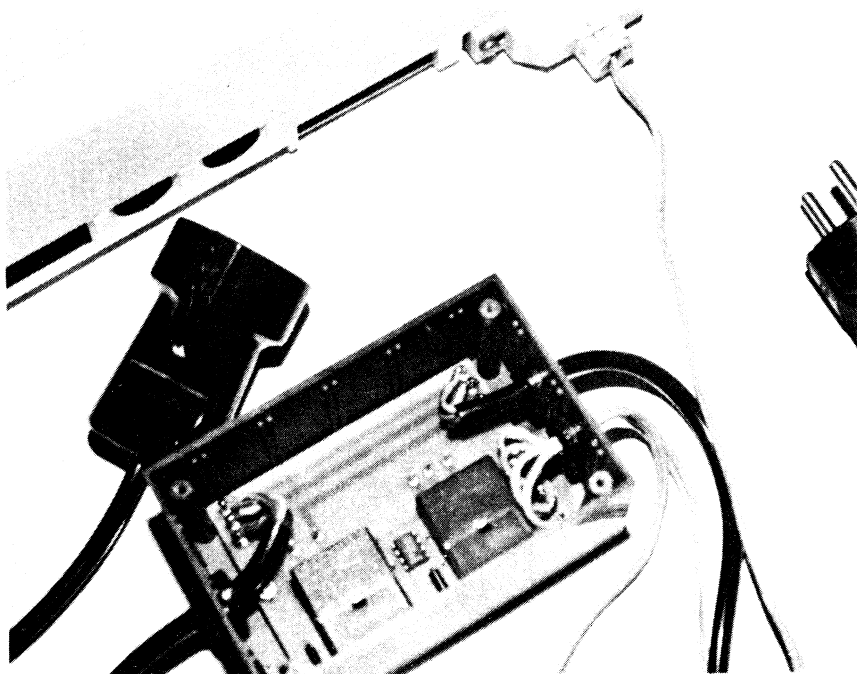


The following picture shows the board and the components.



Be careful while experimenting with the 110 V power!

The next picture shows the interface in an open box. It is recommended that you mount the interface in a box because of the dangerous voltage!



The software necessary to control the device is shown in the following BASIC-program. This program gives you the following control-options:

- 1.) Turn on and off in certain intervalls
- 2.) Turn on and off at a certain time
- 3.) Set clock
- 4.) Turn on
- 5.) Turn off
- 6.) Stop

After you turn on your VIC-20, all lines of port B are input. We can check this with PRINT PEEK (37138). You will see a zero which tells you that all lines are set for data input.

If you want to have all lines set for data output you have to POKE 37138,255. We only use PB0 and control this line with:

```
POKE 37136,1 and
POKE 37136,0
```

If the interface is connected, the POKE command will create a series of audible clicks from the relay.

Note that location 37136 is the data direction register of port B, where you define whether the port is an input or an output and location 37138 is the port itself.

The following program makes it easy to control the relay. It is written for relay 1 but may quickly be changed for relay 2 or expanded for both relays.

```
10 POKE 37138,255
20 GOTO 3000
100 REM TURN ON
110 POKE 37136,0
120 RETURN
200 REM TURN OFF
210 POKE 37136,1
220 RETURN
300 REM SWITCH
```

```

310 IF PEEK(37136)=1 THEN POKE 37136,0:PRINT"ON ":GOSUB 400:PRINT:RETURN
320 IF PEEK(37136)=0 THEN POKE 37136,1
325 PRINT"TURNED OFF";
330 GOSUB 400:PRINT:PRINT:RETURN
400 PRINTLEFT$(TI$,2)":"MID$(TI$,3,2)":"RIGHT$(TI$,2)
410 RETURN
1000 REM ON-OFF EVERY N SECONDS
1010 PRINT"TIMER"
1011 PRINT"ENTER TIME BETWEEN"
1012 PRINT"THE SWITCHING"
1014 INPUT N
1015 GOSUB 300
1020 N=N*60
1030 T=TI+N
1035 GET C$:IF C$<>"" THEN RETURN
1040 IF TI<T THEN 1035
1050 GOSUB 300
1060 GOTO 1030
2000 REM CLOCK FOR SWITCHING
2005 GOSUB 400
2010 PRINT"ENTER TIME"
2020 PRINT"ENTER 000000"
2021 PRINT"WHEN READY"
2030 FOR MX=1 TO 9
2040 PRINT"TIME ";MX
2050 INPUT T$
2060 IF LEN(T$)<>6 THEN PRINT"6 DIGIT IS PLEASE":GOTO 2040
2070 T(MX)=VAL(T$)
2080 IF T(MX)=0 THEN 2100
2090 NEXT MX
2100 FOR J=1 TO MX:PRINT T(J):NEXT J
2110 I=1
2115 PRINT
2130 IF T(I)<VAL(TI$) THEN 2200
2140 PRINT"#####":GOSUB 400
2150 GOTO 2130
2200 GOSUB 300
2210 I=I+1
2220 IF T(I)=0 THEN RETURN
2230 GOTO 2115
2800 PRINT"WHAT TIME IS IT?"
2810 INPUT TIME$:RETURN
2999 PRINT"BYE":END
3000 PRINT"0 PLEASE SELECT"
3010 PRINT"1.TIMER"
3020 PRINT"2.CLOCK"
3030 PRINT"3.SET TIME"
3040 PRINT"4.TURN ON"
3050 PRINT"5.TURN OFF"
3060 PRINT"6.STOP"
3100 GET C$
3110 IF C$="" THEN 3100
3200 C=VAL(C$)
3210 ON C GOSUB 1000,2000,2800,100,200,2999
3220 GOTO 3000

```

Appendix

HOW TO CONNECT UP TO 3 JOYSTICKS TO YOUR VIC-20

This article shows you how to build your own interface, with connections for up to 3 joysticks to your VIC-20 computer. When you buy your computer it has only one port on the right hand side of the case for a single joystick. This port is connected internally with port A of PIA #1 (for details check with the schematics of your computer). One data line of the joystick port is connected with PB7 of port B of PIA #1.

The following section will explain how to connect two additional joysticks to your VIC-20. Your system will then have the capability to operate with a total of three joysticks. At the end of this chapter you will find a program which takes advantage of the additional joysticks. This program even works on the basic version of the VIC-20 with 3.5k of RAM.

Construction of the dual joystick

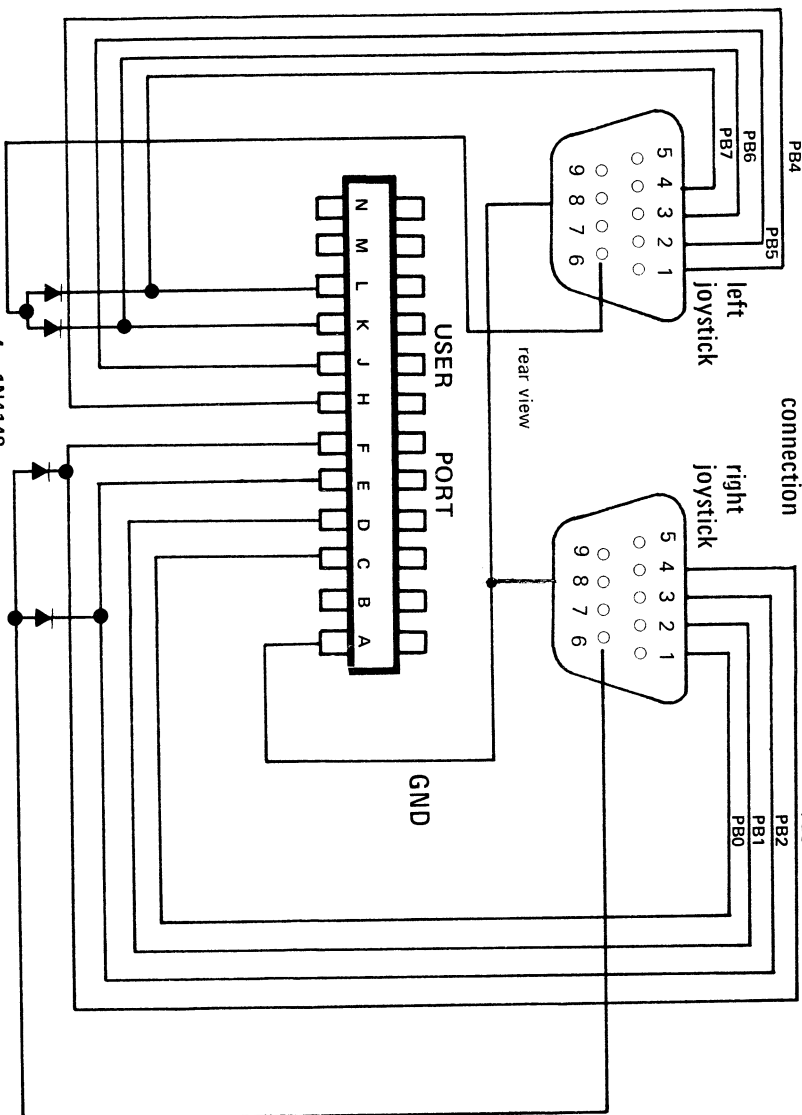
First we have to construct an interface box which plugs into the I/O expander connector at the left hand side of the back of your computer.

List of items required :

- 1 CINCH TRW connector
CARDCON 251-12-50-170
50-24SN-9 8124
- 2 diodes 1N4148 or similar
- 2 joysticks from Commodore or ATARI
- 2 9 pin connectors (see VIC-manual)

Dual joystick for VIC-20

connection



The two joysticks are connected to the user port of the VIC using this interface. The two joysticks are connected to the computer via port B of PIA #1 (6522 VIA chip). After assembling the interface according to the schematics you can plug in the joysticks and test them with the program below :

```
10 POKE 37138,0
15 N=PEEK(37136)
20 PRINT N
25 GOTO 15
```

After you connected the joysticks enter RUN. You will see numerous numbers scrolling in the lefthand side of the screen. The numbers you obtain depends upon the positions of the joystick. The directions are detected as follows :

right joystick :	North	255-1
	South	255-2
	West	255-4
	East	255-8
left joystick :	North	255-16
	South	255-32
	West	255-64
	East	255-128

If you want to write your own programs you also need to know how to detect whether or not the fire-button is pressed. If the firing button of the right joystick is pressed a one is deducted from the actual value. If you push the button on the left joystick, 16 is deducted.

Example: The right joystick supplies a 254 if the button is pushed but no movement in any direction.

Note: If you push the right joystick to north east and push the firing button, you will get 255-1-8=244.

While the program is still running, you should get a value of 243, if the button on the right joystick is depressed. Be aware that the same value will also appear if you move the right joystick to the east and the west simultaneously. This is physically impossible, and the programmer has to rectify this situation. The computer must recognize that the firing button was pressed.

The program has to know that it cannot detect one of the directions as long as the firing button is pressed. To understand this, press the firing button and move the joystick to the east. The value of N should not change at this time. Similar things should happen if you move the joystick to the west. While the program is still running, push the button on the left joystick. The value should now change to 653. The left joystick will behave like the right joystick.

Pressing the firing button has the same effect as a simultaneous movement to east and west. You will get a value of: $N=255-64-128=63$

The program has to know which joystick appears to have moved to the east and to the west at the same time. If this is detected the computer knows that the firing button was pressed.

Now we want to show you a game of skill program which uses the dual joysticks described above. Two players can play against each other. The goal of the game is to run as long as possible around on the screen without hitting the wall or the snake of the opponent player. Moving backward is not allowed. The program works on the basic configuration of the VIC (3,5k RAM).

Player II ---> left joystick
Player I ---> right joystick

Start the program with RUN. You then have a choice between wanting the instructions or the start of the game. The program also allows you to play against the computer.


```

1 PRINT"J":PRINT:PRINT:POKE36879,27
2 PRINTSPC(4);"  VC-SNAKE  ■":GOTO 13
3 PRINT"PRESS 'F' IF YOU WANT TO PLAY"
4 REM
5 PRINT"AGAINST A FRIEND,OR"
6 PRINT"'C' FOR COMPUTER"
7 GOTO 26
8 REM COPYRIGHT C 1982
9 REM WINFRIED HOFACKER
10 PRINT "INSTRUCTIONS ?"
11 GETZZ$
12 IFZZ$="N"THENPRINT:GOTO4
13 IFZZ$<>"Y"THEN14
14 PRINT:PRINT"YOU CAN PLAY THIS"
15 PRINT"GAME AGAINST A FRIEND OR"
16 PRINT "AGAINST THE COMPUTER"
17 GOTO 3
18 GETOP$:IFOP$=""THEN26
19 IFOP$<>"C"ANDOP$<>"F"THEN26
20 IFOP$="F"THEN46
21 PRINT:PRINT"JOYSTICK #1 IS YOURS"
22 PRINT"YOU START IN THE UPPER"
23 PRINT"LEFTHAND CORNER"
24 PRINT:PRINT "PRESS ANY KEY"
25 GET ZZ$: IF ZZ$="" THEN 47
26 DIM DX(4),DY(4),DC(4),MV(15)
27 FORI=1TO4
28 READ DX(I),DY(I),DC(I)
29 NEXT
30 FORN=1TO15:READMV(N):NEXT
31 US(1)=0:US(2)=0:CS=0:N=.92:W=.97:V=1:L=2:R=3:U=4:K=.5
32 GOSUB10000
33 PRINT"■";
34 TL=20:UL(1)=214:UL(2)=219:FT=40:S=32:EN=0:CL=219:CD=2
35 UD(1)=3:UD(2)=2:M=50
36 PZ(1)=7773 :PZ(2)=8092 :CA=8092
37 POKE36879,40
38 TI$="000000"
39 T2=TI
40 ZU=PEEK(37136)
41 PRD=MV((ZUAND240)/16):IFPRD<>0THENU(1)=PRD
42 I=1:GOSUB600:IFEN=1THEN5000
43 IFOP$="C"THEN1000
44 PLD=MV(ZUAND15):IFPLD<>0THENU(2)=PLD
45 I=2:GOSUB600:IFEN=1ANDOP$="F"THEN5000
46 IFEN=1ANDOP$="C"THEN6000

```

```

550 GOTO1900
600 A=PZ(I)+DX(UD(I))+DY(UD(I))
660 IFPEEK(A)<>STHENEN=1:RETURN
750 POKEPZ(I),UL(I)
755 GOSUB 8000
760 POKEA,DC(UD(I))
800 PZ(I)=A
805 RETURN
1000 REM PET MOVE
1030 X=DX(CD):Y=DY(CD)
1040 IF(PEEK(CA+X+X+Y+Y)=S)AND(PEEK(CA+X+Y)=S)THEN1600
1080 IFCD=LORCD=RTHEN1200
1090 IFRND(V)>KTHENH=L:G=R:GOTO1110
1100 H=R:G=L
1110 X=DX(H):Y=DY(H)
1120 IF(PEEK(CA+X+X+Y+Y)=S)AND(PEEK(CA+X+Y)=S)THEN1590
1140 IFH<>GTHENH=G:GOTO1110
1150 GOTO1300
1200 IFRND(V)>KTHENH=V:G=U:GOTO1110
1220 H=U:G=V
1240 GOTO1110
1300 X=DX(CD):Y=DY(CD)
1320 IFPEEK(CA+X+Y)=STHEN1600
1400 IFCD=LORCD=RTHEN1500
1420 IFRND(V)>KTHENH=L:G=R:GOTO1460
1440 H=R:G=L
1460 X=DX(H):Y=DY(H)
1470 IFPEEK(CA+X+Y)=STHEN1590
1480 IFH<>GTHENH=G:GOTO1460
1490 GOTO1590
1500 IFRND(V)>KTHENH=V:G=U:GOTO1460
1520 H=U:G=V
1540 GOTO1460
1590 CD=H
1600 IFRND(V)>WTHEN1080
1650 A=CA+DX(CD)+DY(CD)
1660 IFPEEK(A)<>STHEN6000
1750 POKECA,CL
1760 POKE A,DC(CD)
1800 CA=A
1900 IFTI-TZ<MTHEN1900
1940 M=M*N
1945 IFM<1THEN M=1

```

```

1950 GOTO500
5000 REM CRASH PLAYER
5005 IFI=1THENWW=2
5006 IFI=2THENWW=1
5010 POKEPZ(I),UL(I)
5020 POKEA,DC(UD(I))
5030 US(WW)=US(WW)+1
5035 IFOP$="P"THENC$=C$+1
5040 PRINT"5012PLAYER#"I;
5050 PRINT"PENG! "
5055 GOSUB 7000
5060 PRINT"5012SP. #1:"US(1);
5062 IFOP$="P"THENPRINT"    PET:"C$:GOTO5070
5065 PRINT"PL.#2:"US(2)
5070 TT=TI
5080 GETA$
5090 IFTI-TT<4*60THEN5080
5100 GOTO100
6000 REM CRASH VC-20
6010 POKECA,CL
6020 POKEA,DC(CD)
6030 US(1)=US(1)+1:N=N*.98:IFN<.5THENN=.5
6040 PRINT"5012 VC-20";
6050 GOTO5050
7000 POKE36877,220
7010 FOR P=15 TO 0 STEP-1
7020 POKE 36878,P
7030 FOR H=1TO 10
7040 NEXT H
7050 NEXT P
7060 POKE 36877,0
7070 POKE 36878,0
7080 RETURN
8000 POKE36878,15
8010 POKE36876,220
8020 FOR P=1TO5
8030 NEXT P
8040 POKE36878,0
8050 FOR P=1TO50
8060 NEXT P
8070 POKE36876,200
8080 FOR P=1 TO 5
8090   "











```

```

8100 POKE36876,0
8110 FOR P=1T050
8120 NEXT P
8130 POKE36878,0
8140 RETURN
10000 REM DRAW BORDERS
10010 PRINT"┐";A$="※"
10020 FORI=1T022:PRINTA$;:NEXT
10040 FORI=1T021:PRINTA$;SPC(20);A$;:NEXT
10060 FORI=1T021:PRINTA$;:NEXT:PRINT"┌";RIGHT$(A$,1);
10100 RETURN
11000 ZU=PEEK(37136)
11005 IFZU=251THEND=1:GOTO570
11010 IFZU=247THEND=2:GOTO570
11015 IFZU=239THEND=3:GOTO570
11020 IFZU=253THEND=4:GOTO570
11025 GOTO600
50000 DATA 0,22,20,-1,0,60,1,0,62,0,-22,30
50005 DATA0,0,0,0,0,0,3,0,0,0,2,0,1,4,0

```

Because it is not always easy to identify the control characters in a BASIC listing, we put together a summary of the VIC-20 cursor control characters.

ASCII	PRINTS	DESCRIPTION	
146		Reverse Off	Use CTRL key
18		Reverse On	Use CTRL key
3		RUN / STOP	
147		Clear	CRL
19		Home	
145		Cursor Up	
17		Cursor Down	
157		Cursor Left	
29		Cursor Right	
148		Insert	

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES



ORDER FORM

HOFACKER



ELCOMP PUBLISHING, INC., 53 Redrock Lane, Pomona, CA 91766 (Phone: (714) 623-8314)

Please make checks out to ELCOMP PUBLISHING, INC.

Name:

Card #

Address:

Expiration Date

City / State / Zip:

Master Charge Bank Code

Signature

Qty.	Order No.	Description	Price \$	Qty.	Order No.	Description	Price \$
.....	29	MICROC. HARDWARE HANDB.	14.95	4889	EXPANDING YOUR VIC	14.95
.....	150	CARE AND FEEDING...	9.95	4894	RUNFILL	9.95
.....	151	8K MICROSOFT BASIC	9.95	4896	MINIASSEMBLER	19.95
.....	152	EXP.HANDB. 6502 AND 6802	9.95	6153	LEARN-FORTH	19.95
.....	153	MICROC. APPLICATION NOTES	9.95	6155	POWER FORTH (APPLE)	39.95
.....	154	COMPLEX SOUND	6.95	7022	ATHONA-1, CASS.	19.95
.....	156	SMALL BUSINESS PROGR.	14.90	7023	ATHONA-1, DISK	24.95
.....	158	SECOND BOOK OF OHIO	7.95	7024	ATHONA-1, CARTRIDGE	59.00
.....	159	THE THIRD BOOK OF OHIO	7.95			
.....	160	THE FOURTH BOOK OF OHIO	9.95	7042	EPROM BURNER FOR ATARI	179.00
.....	161	THE FIFTH BOOK OF OHIO	7.95	7043	EPROM BOARD (CARTRIDGE)	29.95
.....	162	GAMES FOR THE ATARI	7.95	7049	ATHONA-2, CASS.	49.95
.....	164	ATARI LEARNING BY USING	7.95	7050	ATHONA-2, DISK	54.00
.....	166	PROGR. I.6502 MACH.LANG.	19.95	7053	LEARN-FORTH	19.95
.....	169	HOW TO PROGR.IN MACH.L.	9.95	7055	POWER FORTH	39.95
.....	170	FORTH ON THE ATARI	7.95	7098	ATAS	49.95
.....	171	A LOOK INTO THE FUTURE	9.95	7099	ATMAS	89.00
.....	173	PROGRAM DESCRIPTIONS	4.95	7100	PROGRAMS FROM BOOK # 164	29.95
.....	174	ZX-81/TIMEX	9.95	7200	INVOICE WRITING,DISK	39.00
.....	176	TRICKS FOR VICS	9.95	7207	GUNFIGHT FOR ATARI	19.95
.....	202	JANA/1 MONITOR	29.95	7210	WORDPROCESSOR, CASS.	29.95
.....	604	PROTOTYPING CARD	29.00	7211	HOW TO CONNECT...	19.95
.....	605	6522 VIA I/O EXP. CARD	39.00	7212	MAILING LIST, CASS.	19.95
.....	606	SLOT REPEATER	49.00	7213	MAILING LIST, DISK	24.95
.....	607	2716 EPROM PROGRAMMER	49.00	7214	INVENTORY CONTR., CASS.	19.95
.....	608	SOUND WITH THE 61 AY3-8912	39.00	7215	INVENTORY CONTR., DISK	24.95
.....	609	8K EPROM CARD (2716)	29.00	7216	WORDPROCESSOR, DISK	34.95
.....	610	12 BIT A/D CONVERTER BOAR	74.00	7217	WORDPROCESSOR, CARTRIDGE	69.00
.....	615	16K RAMROM-BOARD	59.95	7221	PROGRAMS FROM BOOK # 162	29.95
.....	680	THE CUSTOM APPLE BOOK	24.95	7222	KNAUS OGINO	29.95
.....	2398	MAILING LIST FOR ZX-81	19.95	7223	ASTROLOGY PROGRAM	29.95
.....	2399	MACHINE LANG. MONITOR	9.95	7224	EPROM BOARD KIT	14.94
.....	2400	ADAPTER BOARD FOR ZX-81	14.80	7230	FLOATING POINT PACKAGE	299.95
.....	3276	EDITOR/ASS. FOR CBM	39.00	7291	RS232-INTERFACE	19.95
.....	3475	ASSEMBLER FOR CBM	39.95	7292	EPROM BURNER KIT	49.00
.....	4826	GUNFIGHT FOR PET/CBM	9.95	7307	ATCASH	49.95
.....	4844	UNIVERSAL EXP. BOARD	18.95	7309	MOON PHASES	19.95
.....	4848	ADAPTER BOARD	3.95	7310	ATAMMO	29.95
.....	4870	PROFI WORDPROC. F. VIC	19.95	7312	SUPERMAIL	49.00
.....	4880	TIC TAC VIC	9.95	7313	BUSIPACK 1	98.00
.....	4881	GAMEPACK FOR VIC	14.95	7314	BIORHYTHM FOR ATARI	9.95
.....	4883	MAIL-VIC	14.95			

Payment: Check, Money Order, VISA, Mastercharge, Access, Interbank, Eurocheck

Prepaid orders add \$3.50 for shipping (USA)

\$5.00 handling fee for C.O.D.

ALL ORDERS OUTSIDE USA: Add 15 % shipping.

California residents add 6.5% sales tax.

Superboard and CIP are trademarks of Ohio Scientific Co.

ATARI is a trademark of ATARI Warner Communications

PET, CBM and VIC-20 are trademarks of Commodore Business Machines.

TRS-80 is a trademark of TANDY Radio Shack.

APPLE II is a trademark of APPLE Computer Inc.

BITFIRE

BITFIRE

BITFIRE

BITFIRE

BITFIRE

BITFIRE

BITFIRE