PRICE £1-00

ISSUE 4

THE U.K. ATARI COMPUTER OWNERS CLUB

INDEPENDENT USER GROUP

DISPLAY LISTS
TEXT ON GRAPHICS 8
PICK-A-STICK
CONTACT
SPECIAL OFFERS
SOFTWARE LIBRARY

Plus lots of program listings
PECKMAN
STUNT RIDER
SPACE FORTRESS
HEX
and many more!!

THE U.K. ATARI COMPUTER OWNERS CLUB

INDEPENDENT USER GROUP

CONTENTS

2 TEXT ON GRAPHICS 8
Add text to graphs or high resolution pictures you have created.

5 PICK-A-STICK
Comprehensive review of many of the joysticks available.

8 DISPLAY LISTS
Learn all about Display Lists, including scrolling.

12 CONTACT
Your opportunity to meet other ATARI computer nuts.

13 SOFTWARE LIBRARY
Details of your library system and a list of titles available in the "3-for-1" exchange scheme.

16 TOP TEN
A chart of the most requested programs from the library.

16 SOFTWARE MAP
Is your name on the map.

17 INTERFACE
A selection of problems and queries received from members.

18 SPECIAL OFFERS
Lots of software for you to purchase.

19 PECKMAN
BASIC version of a well known arcade game.

21 COMPUTER ASSISTED DESIGN
Have fun designing new shapes.

24 STUNT RIDER
Jump the buses with your motorbike.

25 DID YOU KNOW. . .
Hints, tips and mini-programs.

26 SPACE FORTRESS
Defend your space fortress from the missiles.

27 HEX
Two player game of skill.

SUBSCRIPTION

Welcome to issue 4 of the newsletter. In this issue we have articles on DISPLAY LISTS (including scrolling), and TEXT ON GRAPHICS 8, which shows you how to print text over a hi-res screen. Also in this issue is a review of most of the joysticks available for the ATARI, plus our regular features and lots of exciting program listings.

At the PCW show in the Barbican we saw the new range of ATARI computers and peripherals, which included the three new printers, the CP/M module, the touch tablet (we liked this one!), the new dual density disk drive, the light pen, and a selection of new software, eventually to be supplied by ATARI.

On receiving this issue, many of you will also receive a reminder that your four-issue subscription has expired. For us to be able to maintain the quality of the newsletter, we would like you to re-subscribe for a further four issues. In order to help cover the increased cost of producing a newsletter of this standard, we did intend to raise the membership fee to five pounds, but we are now subsidising the club funds by selling the newsletters at exhibitions and over the counter at MAPLIN shops. We would like to point out however, that non-members who purchase a copy are not only paying more for that issue, but are also not elligible for special offers or the software exchange library.

Finally we would like to thank those members who wrote to us with praise for our efforts on issue three. It is nice to know that our hard work is appreciated.

CREDITS

Editor	Chris Barlow
Technical Editor	Ron Levy
Technical Editor	Keith Mayhew
Software Librarian	Roy Smith
Art Editor	Peter Blackmore
Subscriptions	Liz Robinson
Photography	John Attfield

Send articles, comments, letters and software contributions to:
The U.K. ATARI COMPUTER OWNERS CLUB, P.O. BOX 3, Rayleigh, Essex.

TEXT ON GRAPHICS 8

Have you ever wanted to put text or numerical labels on to a high resolution drawing or graph? If so, then we have just the thing for you! There have been several programs published to mix modes 8 and 0 on the ATARI, but each has had its own annoying operating deficiencies. Some have been written as BASIC subroutines and so are very slow, some have been written in machine code but require USR commands with a lot of parameters to be defined, and nearly all will only allow you to print to the nearest mode 0 character block positions.

Our program, however, allows you to print your characters to any pixel co-ordinate in the X or Y axis. It has also been made very simple to use, since only two commands are needed to print the string (T$) in the required position. These two commands are:—

POSITION X,Y
Z=USR(ADR(TEXT$))

As you can see we have used the POSITION command to define the location to start printing just as it is used in graphics mode 0, but in this case the values range from 0 to 311 for the X axis, and from 0 to 183 for the Y axis.

The machine code program itself actually resides in a BASIC string (TEXT$) and is read in as DATA statements at the beginning of your program. Listing one is the source code which will be translated into DATA statements in listing two.

HOW IT WORKS

The first task of the program is to find the starting memory location of 'T$' and also its length. 'T$' contains the text (or any other characters) to be printed.

In fact the character to be plotted depends on which character set is being used, as you may be using your own redefined character set(s).

FINDING THE STRING

Firstly, T$ is located in the VARIABLE NAME TABLE and its position is noted. Then, by multiplying this position by eight, its sub-table can be found which contains the information for that string. From here we get its length and its offset from the start of the string storage area. Having extracted this information the string is now plotted as follows.

PLOTTING THE STRING

The next task is to calculate the memory location to place the first byte on the screen. Firstly, the cursor row is multiplied by 40 (there are forty bytes per mode 8 line) then added to that is the cursor column number divided by eight. E.g. If the cursor was at row 10 and column 80 then the number is: (10*40)+(80/8)=410.

This number is then added to the starting location of the screen. This result represents the memory location into which the top left byte of the string will be placed.

Next, the top byte needed to form the character shape is obtained from the character set table, a process which has been adequately covered in previous issues and will not be repeated here. Unfortunately, it is not a simple matter of dumping this byte into a memory location, because depending upon the precise position required for this byte, it may not coincide with the nearest byte on the screen. This is a little difficult to grasp, but figure 1 should help to explain the problem.

The answer is to use two temporary storage registers, and to shift the character byte along allowing it to overflow into another byte if necessary. This shifted byte can then have the overflow byte from the previous character mapped onto it (assuming that the text to be printed contains more than one character) and it is then placed onto the screen position allocated. The byte overflowed into from the latest character shift is then moved to the second temporary storage register. See Figure 2.

SEQUENCE OF EVENTS

1. Clear byte 1.
2. Load accumulator with character data.
3. Shift accumulator byte towards byte 1.
4. 'OR' byte 2 onto accumulator.
5. Store accumulator contents to screen.
6. Transfer byte 1 to byte 2.

The program is also faced with a further complication in that we do not want to lose any graphics (eg. plots and drawto's) which are adjacent to the first and last characters. This has also been allowed for, and the program will plot text with great speed and accuracy.

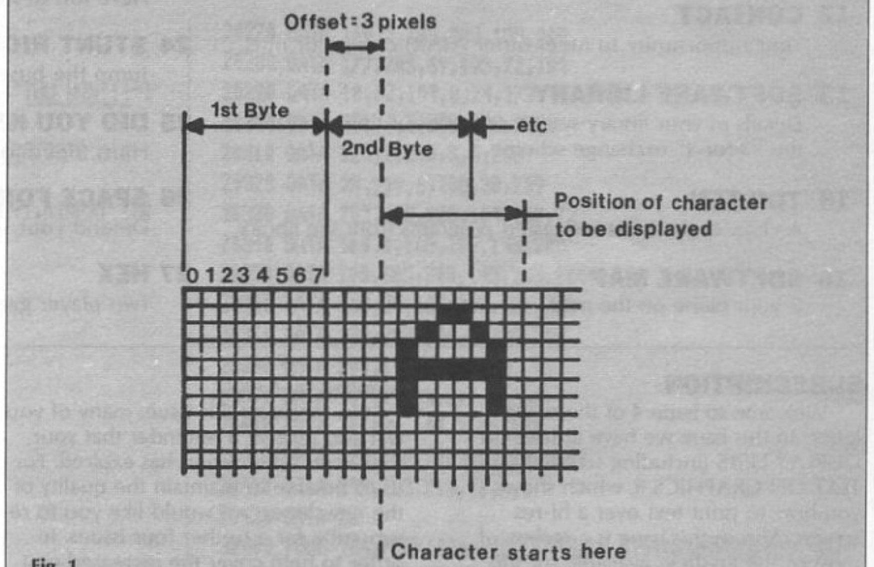


Fig. 1

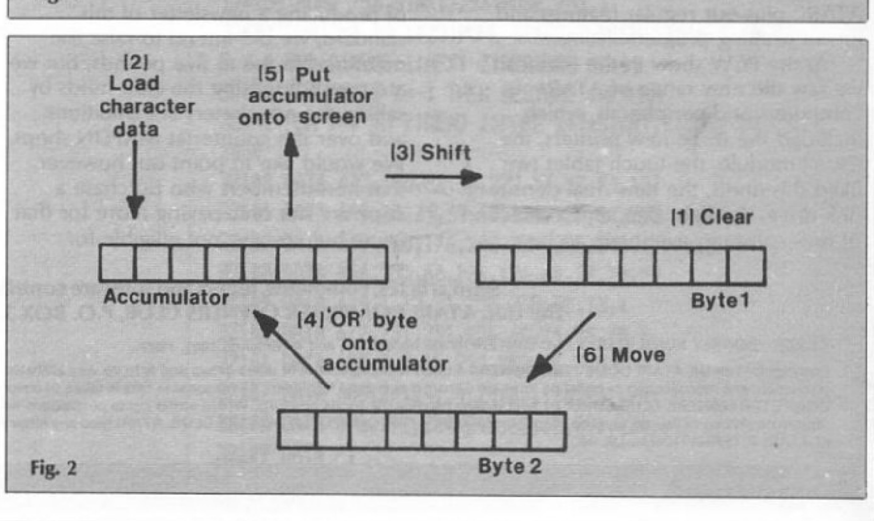


Fig. 2

Listing 1

TEXT ON GRAPHICS 8

```
0100 ; Graphics 8 character plotter
0110 ;  routine for use with BASIC
0120 ;
0130 ;O.S. Equates.
0140 ;
0150 ROWCRS  =    $54
0160 COLCRS  =    $55
0170 SAVMSC  =    $58
0180 CHBAS   =    $02F4
0190 ;
0200 ;BASIC 'strings' equates.
0210 ;
0220 VNTP    =    $82
0230 VVTP    =    $86
0240 STARP   =    $8C
0250 ;
0260 ;Program variables equates.
0270 ;
0280 STRLEN  =    $20
0290 CHRCNT  =    $21
0300 TEMP    =    $22
0310 OFFSET  =    $24
0320 BYTE1   =    $25
0330 BYTE2   =    $26
0340 LINCNT  =    $27
0350 DATBLK  =    $28
0360 STRLOC  =    $2A
0370 CHAR    =    $2C
0380 SCREEN  =    $2E
0390 CHR     =    $CB
0400         *=   $4000
0410         PLA
0420 ;Move variable name table
0430 ;pointer (VNTP) to our page
0440 ;zero pointer  (CHR).
0450         LDA  VNTP
0460         STA  CHR
0470         LDA  VNTP+1
0480         STA  CHR+1
0490 ;'X' is used for string count.
0500         LDX  #$00
0510 NXNAM   LDY  #$FF
0520 NXCHR   INY
0530         LDA  (CHR),Y
0540         BPL  NXCHR
0550         INX
0560         CPY  #$01
0570         BNE  INCNAM
0580         CMP  #'$'
0590         BNE  INCNAM
0600         DEY
0610         LDA  (CHR),Y
0620         INY
0630         CMP  #'T'
0640         BNE  INCNAM
0650         BEQ  FNDNAM
0660 ;Points to next variable name.
0670 INCNAM  LDA  CHR
0680         INY
0690         STY  TEMP
0700         CLC
0710         ADC  TEMP
0720         STA  CHR
0730         LDA  CHR+1
0740         ADC  #$00
0750         STA  CHR+1
0760         LDA  #$00
0770         BEQ  NXNAM
0780 ;Found position in name table.
0790 ;
0800 ;Now find string length and
0810 ; start address.
0820 FNDNAM  DEX
0830         TXA
0840         LDX  #$00
0850         STX  DATBLK+1
0860         CLC
0870         ROL  A
0880         ROL  DATBLK+1
0890         ROL  A
0900         ROL  DATBLK+1
0910         ROL  A
0920         ROL  DATBLK+1
0930         CLC
0940         ADC  VVTP
0950         STA  DATBLK
0960         LDA  DATBLK+1
0970         ADC  VVTP+1
0980         STA  DATBLK+1
0990         LDY  #$02
1000         LDA  (DATBLK),Y
1010         STA  STRLOC
1020         INY
1030         LDA  (DATBLK),Y
1040         STA  STRLOC+1
1050         INY
1060         LDA  (DATBLK),Y
1070         STA  STRLEN
1080         LDA  STRLOC
1090         CLC
1100         ADC  STARP
1110         STA  STRLOC
1120         LDA  STRLOC+1
1130         ADC  STARP+1
1140         STA  STRLOC+1
1150 ;Found string and length.
1160 ;
1170 ;Now add column and row to
1180 ; the screen start address.
1190         LDA  ROWCRS
1200         STA  SCREEN
1210         STA  TEMP
1220         LDA  #$00
1230         STA  SCREEN+1
1240         STA  TEMP+1
1250         LDX  #$05
1260 MUL1    CLC
1270         ROL  SCREEN
1280         ROL  SCREEN+1
1290         DEX
1300         BNE  MUL1
1310         LDX  #$03
1320 MUL2    CLC
1330         ROL  TEMP
1340         ROL  TEMP+1
1350         DEX
1360         BNE  MUL2
1370         LDA  SCREEN
1380         CLC
1390         ADC  TEMP
1400         STA  SCREEN
1410         LDA  SCREEN+1
1420         ADC  TEMP+1
1430         STA  SCREEN+1
1440         LDA  SCREEN
1450         CLC
1460         ADC  SAVMSC
1470         STA  SCREEN
1480         LDA  SCREEN+1
1490         ADC  SAVMSC+1
1500         STA  SCREEN+1
1510         LDA  COLCRS
1520         STA  TEMP
1530         AND  #$07
1540 ;   ('OFFSET' contains display
1550 ;      horizontal bit offset.)
1560         STA  OFFSET
1570         LDA  COLCRS+1
1580         STA  TEMP+1
1590         LDX  #$03
1600 DIV1    CLC
1610         ROR  TEMP+1
1620         ROR  TEMP
1630         DEX
1640         BNE  DIV1
1650         LDA  SCREEN
1660         CLC
1670         ADC  TEMP
1680         STA  SCREEN
1690         LDA  SCREEN+1
1700         ADC  TEMP+1
1710         STA  SCREEN+1
1720 ;Reset our scan line counter.
1730         LDA  #$00
1740         STA  LINCNT
1750 ;Start of main program loop.
1760 OTLOOP  LDA  OFFSET
1770         EOR  #$07
1780         TAX
1790         STX  TEMP
1800         LDY  #$00
1810         LDA  (SCREEN),Y
1820 SHF1    LSR  A
1830         DEX
1840         BPL  SHF1
1850         LDX  TEMP
1860 SHF2    ASL  A
1870         DEX
1880         BPL  SHF2
1890         STA  BYTE2
1900 ;Reset character count.
1910         LDA  #$00
1920         STA  CHRCNT
1930 ;Scan line plotting loop.
1940 INLOOP  LDY  CHRCNT
1950         LDA  (STRLOC),Y
1960         AND  #$7F
1970         CMP  #32
1980         BCS  N1
1990         CLC
2000         ADC  #64
2010         LDY  #$00
```

```
2020          BEQ    N2
2030 N1       CMP    #96
2040          BCS    N2
2050          SEC
2060          SBC    #32
2070 N2       STA    CHAR
2080          LDA    #$00
2090          STA    CHAR+1
2100          STA    BYTE1
2110          LDX    #$03
2120 MUL3     CLC
2130          ROL    CHAR
2140          ROL    CHAR+1
2150          DEX
2160          BNE    MUL3
2170          LDA    CHAR+1
2180          CLC
2190          ADC    CHBAS
2200          STA    CHAR+1
2210          LDY    LINCNT
2220          LDA    (CHAR),Y
2230          LDX    #$FF
2240 LOOP1    INX
2250          CPX    OFFSET
2260          BEQ    NXTCHR
2270          CLC
2280          ROR    A
2290          ROR    BYTE1
2300          LDY    #$00
2310          BEQ    LOOP1
2320 SKIP     BEQ    OTLOOP
2330 ;Insert RH pixels on end byte.
2340 NXTCHR   ORA    BYTE2
2350          LDY    CHRCNT
2360          STA    (SCREEN),Y
2370          LDA    BYTE1
2380          STA    BYTE2
2390          INC    CHRCNT
2400          LDA    CHRCNT
2410          CMP    STRLEN
2420          BNE    INLOOP
2430          LDY    CHRCNT
2440          LDA    (SCREEN),Y
2450          LDX    OFFSET
2460          BEQ    SHFT
2470 SHF3     ASL    A
2480          DEX
2490          BNE    SHF3
2500          LDX    OFFSET
2510 SHF4     LSR    A
2520          DEX
2530          BNE    SHF4
2540 SHFT     ORA    BYTE2
2550          STA    (SCREEN),Y
2560          LDA    SCREEN
2570          CLC
2580          ADC    #40
2590          STA    SCREEN
2600          LDA    SCREEN+1
2610          ADC    #$00
2620          STA    SCREEN+1
2630          INC    LINCNT
2640          LDA    LINCNT
2650          CMP    #$08
2660          BEQ    RETURN
2670          LDY    #$00
2680          BEQ    SKIP
2690 RETURN   RTS    ;Return to BASIC.
```

USING FROM BASIC

After converting the machine code program into DATA statements, this data is then added to a few basic instructions, see listing 2. TEXT$ is DIMensioned to 374 because the machine code program has been converted to 374 DATA statements.

T$ has been set to 40 for convenience, but ideally should be set to a number equivalent to the longest block of characters you wish to print on the screen. Line 20 actually READs the DATA and puts it, one byte at a time, into TEXT$.

When you have typed in listing 2 you should LIST this to a cassette or disk file, i.e. LIST "C:" or LIST "D:FILENAME". This is so that you can ENTER it into an existing program at a later date by typing either ENTER "C:" or ENTER "D:FILENAME".

Listing 2

```
10 DIM TEXT$(374),T$(40)
20 FOR I=1 TO 374:READ T:TEXT$(I,I)=CHR$
(T):NEXT I
32700 REM ..... DATA FOR M/C SUBROUTINE
.....
32701 DATA 104,165,130,133,203,165,131,1
33
32702 DATA 204,162,0,160,255,200,177,203
32703 DATA 16,251,232,192,1,208,14,201
32704 DATA 164,208,10,136,177,203,200,201
32705 DATA 84,208,2,240,20,165,203,200
32706 DATA 132,34,24,101,34,133,203,165
32707 DATA 204,105,0,133,204,169,0,240
32708 DATA 210,202,138,162,0,134,41,24
32709 DATA 42,38,41,42,38,41,42,38
32710 DATA 41,24,101,134,133,40,165,41
32711 DATA 101,135,133,41,160,2,177,40
32712 DATA 133,42,200,177,40,133,43,200
32713 DATA 177,40,133,32,165,42,24,101
32714 DATA 140,133,42,165,43,101,141,133
32715 DATA 43,165,84,133,46,133,34,169
32716 DATA 0,133,47,133,35,162,5,24
32717 DATA 38,46,38,47,202,208,248,162
32718 DATA 3,24,38,34,38,35,202,208
32719 DATA 248,165,46,24,101,34,133,46
32720 DATA 165,47,101,35,133,47,165,46
32721 DATA 24,101,88,133,46,165,47,101
32722 DATA 89,133,47,165,85,133,34,41
32723 DATA 7,133,36,165,86,133,35,162
32724 DATA 3,24,102,35,102,34,202,208
32725 DATA 248,165,46,24,101,34,133,46
32726 DATA 165,47,101,35,133,47,169,0
32727 DATA 133,39,165,36,73,7,170,134
32728 DATA 34,160,0,177,46,74,202,16
32729 DATA 252,166,34,10,202,16,252,133
32730 DATA 38,169,0,133,33,164,33,177
32731 DATA 42,41,127,201,32,176,7,24
32732 DATA 105,64,160,0,240,7,201,96
32733 DATA 176,3,56,233,32,133,44,169
32734 DATA 0,133,45,133,37,162,3,24
32735 DATA 38,44,38,45,202,208,248,165
32736 DATA 45,24,109,244,2,133,45,164
32737 DATA 39,177,44,162,255,232,228,36
32738 DATA 240,10,24,106,102,37,160,0
32739 DATA 240,243,240,158,5,38,164,33
32740 DATA 145,46,165,37,133,38,230,33
32741 DATA 165,33,197,32,208,167,164,33
32742 DATA 177,46,166,36,240,10,10,202
32743 DATA 208,252,166,36,74,202,208,252
32744 DATA 5,38,145,46,165,46,24,105
32745 DATA 40,133,46,165,47,105,0,133
32746 DATA 47,230,39,165,39,201,8,240
32747 DATA 4,160,0,240,189,96
```

DEMO GRAPH

Listing 3 is a demonstration program which draws SINE and COSINE waveforms and then labels the coordinates. As you can see, this program incorporates the POSITION X,Y and Z=USR(ADR(TEXT$)) functions mentioned earlier. Once you have typed this in, you should ENTER over it your copy of listing 2, before typing RUN.

Listing 3

```
30 GRAPHICS 24:POKE 710,196:POKE 712,196
:POKE 709,15
40 COLOR 1:DEG :X=160:Y=96
50 PLOT 0,Y:DRAWTO 319,Y
60 PLOT X,0:DRAWTO X,191
70 PLOT 0,Y
80 FOR T=-360 TO 360 STEP 10:GOSUB 500
90 DRAWTO X+(T*(159/360)),Y-Y1
110 NEXT T
120 PLOT 0,0
130 FOR T=-360 TO 360 STEP 10:GOSUB 500
140 DRAWTO X+(T*(159/360)),Y-Y2
150 NEXT T
160 T$="1"
170 POSITION X-12,0:Z=USR(ADR(TEXT$))
180 T$="-1"
190 POSITION X-20,183:Z=USR(ADR(TEXT$))
200 T$="-360"
210 POSITION 0,Y+2:Z=USR(ADR(TEXT$))
220 T$="-180"
230 POSITION 68,Y+2:Z=USR(ADR(TEXT$))
240 T$="0"
250 POSITION 162,Y+2:Z=USR(ADR(TEXT$))
260 T$="180"
270 POSITION 228,Y+2:Z=USR(ADR(TEXT$))
280 T$="360"
290 POSITION 295,Y+2:Z=USR(ADR(TEXT$))
300 T$="d(SIN x)"
310 POSITION 253,34:Z=USR(ADR(TEXT$))
315 T$="--------"
316 POSITION 253,42:Z=USR(ADR(TEXT$))
317 T$="dx"
318 POSITION 253,50:Z=USR(ADR(TEXT$))
320 T$="SIN x"
330 POSITION 50,50:Z=USR(ADR(TEXT$))
499 GOTO 499
500 Y1=SIN(T)*95:Y2=COS(T)*95
510 RETURN
```

PICK-A-STICK

by Derek Dodson

In this review I will be looking at ten joysticks, including the Atari Joystick. I wanted this review to be as unbiased as possible and to this end I wish to thank my wife for her opinions and help in making final decisions.

For the review I needed some progams that would put the different units through their paces. The criteria for selection boiled down to multiple direction changes, and the ease and speed of selecting those directions, comfort of use over protracted periods and, of course, the use of the fire button.

The programs finally selected for the exercise were Jawbreaker and Missile Command. Jawbreaker because, although not using the fire button, requires rapid 90 degree turns, and Missile Command because it utilises all eight directions, rapid use of the fire button and again fast direction changes. Lastly I believe that most people who own Atari Computers would be familiar with Missile Command and Jawbreaker (PAC-MAN), also these two games should put enough stress into both player and joystick, that if a joystick is going to break (or for that matter a player) these are the ones to do it!

At the end of each review I will sum up the findings in the form of a chart awarding points for COMFORT OF USE, ACTION, LOOKS, CONSTRUCTION and finally VALUE FOR MONEY. All prices quoted are approximate only as they obviously vary from outlet to outlet. The rest is up to you . . .

1 Star - POOR 3 Star - GOOD
5 Star - EXCELLENT

POINTMASTER by DISCWASHER (tm) £15.00

The length of lead on this joystick is 63 inches as opposed to 50 on the Atari. The first impression of this unit was that it was cheap in both construction and looks. The fire button is mounted on the grip facilitating both left and right handed use. In use, I found that although the base affords a good grip via a well on the top, this same well has some uncomfortable ridges inside arranged as sort of compass points. The fire button was comfortable and responsive, but for movement the grip had to be moved some distance to make the internal contacts. These contacts are of the same type of construction as in the Atari, i.e. a circuit board with little sprung cups, which are pressed onto the circuit board by a plastic insert mounted in the hand grip. This same insert looks as if it could weaken and bend, making movement less responsive with the passage of time.

COMFORT	★ ★
ACTION	★
LOOKS	★
CONSTRUCTION	★ ★
VALUE	★

COMPETITION-PRO by COIN CONTROLS inc £16.50

Lead length on this joystick is 59 inches. The first impression when confronted by this unit is "SOLID". This is certainly rugged even though it's at the expense of looks, which are a little ugly. There are two large fire buttons for use by left or right hand. The grip is a comfortable ball which you can really grab, in fact it almost pleads to be grabbed. Unfortunately, the base of this one is lacking on comfort and indeed is awkward to hold over a long period. The internal construction is well laid out and uses spring metal contacts all round. I doubt if even the most aggresive use of the joystick would cause it any harm and the action in use was found to be positive and responsive. I have been informed that this unit carries a five year warranty in the States but it only carries the normal warranty over here, the reason for which I do not know. This unit is being distributed in the U.K. by KEMPSTON.

COMFORT	★ ★
ACTION	★ ★ ★ ★
LOOKS	★ ★
CONSTRUCTION	★ ★ ★ ★ ★
VALUE	★ ★

ATARI JOYSTICK by ATARI £7.50

Lead length is a short 50 inches. To look at, this joystick is of quite a sturdy construction and is not unpleasant in appearance. There is however only one fire button on this unit and that is mounted on the left of the base for right handed use. A nice point on this one, is a rubber housing covering the grip which marries up to the base, affording protection from dirt and to a certain extent liquid. The base is fairly comfortable to hold and the grip, although not really long enough to fill ones hand, is quite comfortable if used like a toggle, the sort you find on radio control units. In use it is responsive if not positive, diagonal movement is easy to find when you don't want it, as is straight movement. The fire button is positive and internal construction is of the circuit board and spring cup type. I understand that replacement parts are available for some of the items prone to wear in the unit. Spares for the other units mentioned may be available.

COMFORT	★ ★
ACTION	★ ★ ★
LOOKS	★ ★
CONSTRUCTION	★ ★ ★
VALUE	★ ★ ★ ★ ★

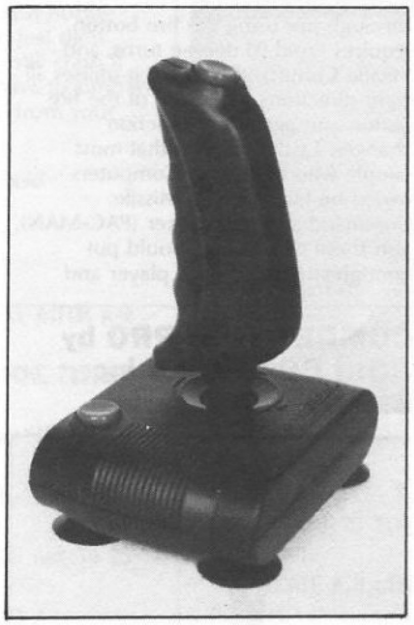
VIDEO COMMAND by ZIRCON INTERNATIONAL inc £17.00

Well now, this is a strange one, it looks very much like a rip-off of the better known 'LE-STICK', except that 'looks' is where the similarity ends. The unit sports an impressive 74 inch lead which comes in very handy if, like me, you sometimes have a few friends gathered about your Atari, waiting for their turn to zap some weird and wonderful foe, and apart from almost sitting on each others lap, when you pass the joystick over the lead either ends up stretched across your throat or is pulled from the consul port, the latter being by far the worst as there is blind panic because your aliens are slaughtering you, and you can't fight back. But to continue . . . The Video Command is comfortable to hold and it incorporates the fire button in with the direction handle on top of the grip. This handle is pushed down to fire and back and forth, etc. for movement. In actual use it was found to be rather confusing, especially in the midst of battle. If anything, the unit is rather too sensitive, as I more often than not ended up going diagonal, when I wanted to go straight, and vice versa. Perhaps with practice this could be overcome.

In construction the unit seems to be quite solid but internally the unit uses the same contacts that select direction to keep the handle centered. Part of the contacts have a thin film of plastic insulation which is intended to prevent shorting, when this centralisation is carried out, but the film looks like it could wear through very easily thus causing all sorts of direction problems. I would point out though that no real tests for durability have been carried out on any of these joysticks and so my fears may be unfounded.

COMFORT	★ ★ ★ ★
ACTION	★ ★
LOOKS	★ ★ ★
CONSTRUCTION	★ ★
VALUE	★

QUICK SHOT by SPECTRAVISION £16.00



Short lead on this one, only 48 inches. Mind you, leads are not the be-all and end-all of a joystick. If looks were the only criteria I think this would be the first one I would pick up from a selection. The unit is pleasing to look at and appears to be of solid construction with a shaped grip and comfortable feel all round. This joystick comes with optional suction cups which can be fixed to the base in place of the normal anti-slip pads. I must say that I was very surprised by the power of these cups to adher to even a fairly rough surface. Obviously the smoother the surface the better but I stuck the unit to the wooden arm of my armchair and it stayed put when putting it through its paces. If you happen to be left handed then you should bear these points in mind: the unit does not have much material on the base for a left handed person to hold, so would be restricted to using the suction cups only, also if held in the hand you would have to use the grip mounted fire button. Right handed people take note that there is plenty of base material to hold the unit comfortably and there is a fire button on the base.

In operation this was found to be reasonable on response but the contacts make a noise which sounds like an all too familiar crack of plastic. I am sure you have all had that moment when playing some game or other and you hear a soul-tearing crack come from your joystick, and you freeze as you say to yourself "Oh my God, what have I done." Well, I found myself saying it all the time when using this one.

If you don't suffer from acute paranoia then all's well and good, because it is just the contacts clicking. Internally the unit uses the apparently popular spring cups mounted on a circuit board as contacts, though why they are so noisy in this joystick beats me.

COMFORT	★ ★ ★ ★ ★
ACTION	★ ★
LOOKS	★ ★ ★ ★ ★
CONSTRUCTION	★ ★ ★
VALUE	★ ★ ★

KRAFT JOYSTICK by KRAFT SYSTEMS CO. £14.00

Would you believe a full 96 inch lead on this one! You would not have problems passing this around between your friends.

Looking at this joystick you could be forgiven for thinking that it is not only fragile but very awkward to use, but in actuality this unit cannot be judged by looks as it does seem to be rather sturdy in use. If it is used, using the grip as you would hold a pen, it is surprising how positive and responsive it is. The movement requires little pressure and selection of all eight possible directions is very definite. Two base-mounted fire buttons facilitate the use by both left and right handed players. The fire buttons, as with the grip, are on the small side but this is not unduly apparent in use. The only thing I do not like about this joystick is that it is a sealed unit, having no user-bodgeable parts, should it go wrong. This point also makes it impossible for me to appraise the internal construction.

COMFORT	★ ★
ACTION	★
LOOKS	★
CONSTRUCTION	SEALED UNIT
VALUE	★

PICK-A-STICK

LE STICK by DATASOFT inc £21.00

A mere 38 inch lead here, the shortest of all so far. For those of you who have not heard of this unit before, LE STICK is a single handed joystick which detects direction required according to which way you tilt it. This is achieved by the use of four mercury switches inside the grip. A fire button is mounted on the top and if you squeeze the grip it will quote "Freeze Motion". This is, I feel, an unfortunate use of wording by the manufacturer, as it implies that it will pause the game while you scratch your nose or make tea! What is actually meant by the "Freeze Motion" claim, is that for as long as you are squeezing the grip it does not matter which way you tip the unit as it has in effect been disconnected from the consul I fail to see the real benefit of this feature because if you lay the stick down you realise this switch and the cursor/player goes whizzing off in the direction in which you laid down the unit.

The grip is very comfortable though and, of course, can be used by left and right handed players. The unit is more or less sealed, so is not easily serviceable, should the need arise. I should point out that short of jumping on the unit, the Mercury switches should operate without trouble after all else has failed.

I have tried to master this unit but unless you wish to put in hours of practice, I think that, like me, you will put it down in frustration. Not only is it possible to over-tilt the unit, mucking up the movement, it is all too easy to get the wrong direction because you have not tilted your hand just right. It looks nice and impressive, but I was not impressed, especially by the price tag it carries.

COMFORT	★★★★★
ACTION	★
LOOKS	★★★
CONSTRUCTION	★★★★
VALUE	★★

SLIK STICK by SUNCOM inc. £10.00

All of the next three joysticks are by SUNCOM inc, and have a lead length of 62 inches, which is a reasonable length. This unit, the cheapest of the three, is of quite solid appearance and pleasant to the eye. The unit has one fire button mounted on the base, so is for right handed use only. The base is comfortable to hold and the grip, although small, is adequate, having a ball mounted on the top. In use it was found to be both positive and responsive. I found however that the fire button was a little awkward in so far as it has to be pushed firmly home to function properly. Internally, the unit uses a very rugged method of contact. This comprises a steel ball on the end of the handle and this sits inside a plastic housing, moulded into the base. When the handle is moved the ball touches contacts mounted on the inside of this housing. In fact all three of Suncom's joysticks use the same method and apart from being original it looks virtually indestructable.

COMFORT	★★★
ACTION	★★★★
LOOKS	★★
CONSTRUCTION	★★★★★
VALUE	★★★★★

DID YOU KNOW...

CORRUPTION

Now for another little 'hot tip'. Have you ever written a program which needs to examine a block of raw data such as memory or a disk file in search of small chunks of text? If you have, then you almost certainly will have had control characters corrupting the display as they appeared. Fear not — for there is a simple remedy!

All you have to do is 'POKE 766,1' beforehand, and all control characters will be printed to the screen as their graphic symbols. To return to normal, simply press 'SYSTEM RESET' or 'BREAK' or use 'POKE 766,0' if in a program.

STAR FIGHTER by SUNCOM £14.00

It won't take long to review this one . . . Apart from £4.00 and a straight handle it is completely identical in construction and use as the SLIK STICK.

COMFORT	★★
ACTION	★★★★
LOOKS	★★
CONSTRUCTION	★★★★★
VALUE	★★★

TAC 2 by SUNCOM £19.00

This is the top of the SUNCOM range and is altogether larger. The unit has two fire buttons, approx. twice the size of the Slik Stick and Starfighter. These are both base mounted for left and right handed use and, being larger, seem to over-come the slight inaccuracy of the two lower priced units. The grip is a very rugged ball type with a steel shaft. In use the grip was very comfortable, as was the base. Movement, however, was a trifle clumsy and I attribute this to both the larger size and newness of the unit, so the unit may well loosen up with use. Internal construction is the same as the other two but, of course, with wiring for the larger two buttons taking up the extra space.

COMFORT	★★
ACTION	★★
LOOKS	★★★★
CONSTRUCTION	★★★★★
VALUE	★

DISPLAY LISTS

by Keith Mayhew & Roy Smith

One of the many powerful features of the ATARI computers is the ability to define your own screen display format, called the DISPLAY LIST.

To fully understand how to implement display lists and what is happening inside the machine, it is necessary to have a fuller explanation of the operations carried out by the two integrated circuits (chips!) GTIA and ANTIC.

ANTIC

ANTIC is a true microprocessor, as powerful in its own way as the 6502 which is the main microprocessor used in the ATARI and many other micros. This means that all screen display operations are truly flexible and programmable by the user, in any language including BASIC, unlike all other micros which only use a dedicated video chip to produce displays. The flexibility of ANTIC can be demonstrated by the fact that you can mix many graphics modes to create interesting screen images.

Of course, even when you only call an ordinary graphics mode or even a text mode the operating system of the computer runs its own display building program and stores the data for the display format somewhere in memory, then informs ANTIC where to find this program. ANTIC takes this program step by step, processes it and sends it to GTIA. This whole process to create a display is the same whether you wish to have a straight mode or custom mixed display.

GTIA

GTIA is basically an interface between the digital computer and the analogue display device (that's a T.V.!!!).

When it receives the information from ANTIC it converts the info into a signal to be passed to a television, but before it does this it adds colour and intensity information to the picture bringing it to life. GTIA also creates the well known player/missile graphics or sprites which are superimposed onto any other graphics/text modes which are currently being displayed, and noting any conflicting areas (collision detection).

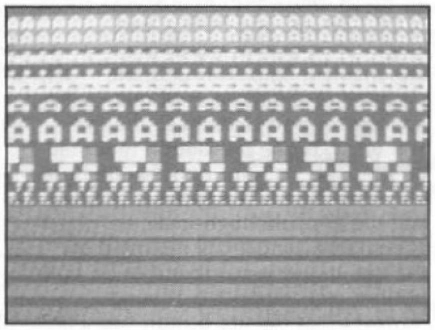
THE TELEVISION

To create interesting images on the T.V. screen by the computer it is important to know how the T.V. itself makes the picture we see. The picture is created by an electron beam fired at the phosphorus coating on the inside of the front screen.

Colour and brightness depend on where the beam hits the screen and how intense the beam is at that point.

The point of light starts at the very top left corner of the screen and moves horizontally to the right.

This single horizontal movement is called a SCAN LINE. The beam then switches off and returns to the left hand side but one scan line lower down and then turns on again to repeat the process. The switched off period when the beam is returning to the left is called the HORIZONTAL BLANK. When the complete picture has been built, line by line in this way, the beam is at the lower right hand corner, it switches off and returns to the upper left corner to start again. This period is called the VERTICAL BLANK. The whole screen is re-drawn fifty times every second to produce a stable image to the eye.



DISPLAY LISTS

When writing a display list you have to call up set numbers of scan lines, dependent on the mode you wish to display, these are called MODE LINES.

For example a mode line in graphics 2 would be sixteen scan lines whereas a mode line in graphics 8 would only consist of a single scan line (see table 1 below).

BASIC MODE	ANTIC MODE	SCAN LINES
0	2	8
1	6	8
2	7	16
3	8	8
4	9	4
5	10	4
6	11	2
7	13	2
8	15	1
9	15	1
10	15	1
11	15	1

Table 1.

When we talk about modes we do not mean modes from BASIC i.e. GRAPHICS 4. We mean ANTIC modes which do not tie up directly with the BASIC modes, in fact, there are more modes supported by ANTIC than BASIC.

So to create a complete display you must stipulate in the program every type of mode line you require, starting at the very top of the screen working down as far as you wish to go. Note that you can in fact create a display list to only cover small sections of the screen by filling in the gaps with blank scan lines. Once the program has run and drawn the required screen it waits until the next vertical blank and then runs the program again.

The actual display list only creates the 'background' of the display.

You have to store the data to be displayed on the mode lines separate to the display list, and then call up that information and place it in the desired position. Within the hardware, there is also the facility to add additional instructions to the display list to allow horizontal and vertical scrolling of a particular mode line, also there is the "LOAD MEMORY SCAN' (LMS) instruction which tells ANTIC where to find the start of screen data. There is a fourth instruction, which enables the use of an interrupt after a mode line has been displayed, more about this later. All of these instructions except 'interrupt' can only be added to mode lines which display data e.g. text and graphics.

GRAPHICS 0 DISPLAY LIST

To start we will give a simple example, showing how ANTIC produces BASIC's GRAPHICS 0 page. Here is the complete display list:

```
0  112
1  112
2  112
3  66
4  64
5  156
6  2
Lines 7 to 27 are all two.
28  2
29  65
30  32
31  156
```

Taking the display list line by line:

In lines 0 to 2 the 112's represent 8 blank scan lines each, thus producing the 24 blank lines at the top of the display (in the same colour as the background). The blue section of the display consists of 24 lines of mode 2, but if you count the number of two's in the list you will only find 23 of them. This is because line 3 is really the first mode 2 line with an instruction added onto it (+64), which is the LMS instruction to tell ANTIC where to find the screen information. Following line 3 there are two extra bytes which are the data for the LMS instruction (lines 4 & 5). In this case the numbers are 64 and 156, but will vary depending on the amount of RAM. To work out the actual address you multiply 156 by 256 and add 64 which equals 40,000, which means the display data for a 48K machine starts at 40,000 in memory. When ANTIC encounters the LMS byte it loads its internal pointer with this value, and then as it displays each mode line it increments this pointer by one to find the next piece of data until it has completed one line (in the case of a mode 2 line this pointer is incremented 40 times per line). Unless ANTIC finds another LMS it will continue incrementing the old value in its pointer, if it does find another LMS (in this case it doesn't) it will reload it with the subsequent value and continue from there. To finish the display list the 'JUMP ON VERTICAL BLANK' (JVB) instruction, 65 on line 29 is implemented, and jumps after vertical blank has occurred to the value found in the next two bytes. This value is worked out in exactly the same way as for lines 4 & 5. E.g. multiply 156 by 256 and add 32 which equals 39968. At location 39968 (on a 48K machine) is the start of the display list i.e. the first line which is 112.

To find out where your graphics 0 display list resides in memory multiply the contents of location 561 by 256 and add the contents of location 560. This will give you the address of the first byte of the display list. To print out the thirty two bytes making up the display list, type in the following program.

```
10 GRAPHICS 0
20 DLIST=PEEK(560)+256*PEEK(561)
30 FOR I=0 TO 31
40 PRINT I,PEEK(DLIST+I)
50 NEXT I
```

CREATING A DISPLAY LIST

As mentioned earlier there are more ANTIC modes than BASIC modes, table 2 shows all the ANTIC modes and instruction values etc.

DISPLAY LISTS

	ANTIC MODES	BASIC MODE	HEX	DEC	
Character Modes		0	02	2	You can add 'HS', 'VS', 'LMS', or 'INT' to all these codes.
		–	03	3	
		–	04	4	
		–	05	5	
		1	06	6	
		2	07	7	
Graphic Modes		3	08	8	
		4	09	9	
		5	0A	10	
		6	0B	11	
		–	0C	12	
		7	0D	13	
		–	0E	14	
		8	0F	15	

BLANK LINES	HEX	DEC	
1	00	0	You can add 'INT' only to these codes.
2	10	16	
3	20	32	
4	30	48	
5	40	64	
6	50	80	
7	60	96	
8	70	112	

INSTRUCTION CODES	HEX	DEC
JMP	01	1
JVB	41	65
HS	10	16
VS	20	32
LMS	40	64
INT	80	128

Table 2.

Now you have the table to refer to, we shall create a display list which will incorporate all the ANTIC display modes from '2' to '15' including blank lines. The display list we create could be stored in any free area of memory, but for convenience, we shall place it on page six, i.e. 1536 onwards.

Listing 1

```
10  FOR I=0 TO 51:READ D
20  POKE 1536+I,D:NEXT I
30  POKE 560,0:POKE 561,6
40  S=156*256
50  FOR I=0 TO 199
60  POKE S+I,33:NEXT I
70  FOR I=200 TO 399
80  POKE S+I,27:NEXT I
90  FOR I=400 TO 759
100 POKE S+I,0:NEXT I
110 GOTO 110
120 DATA 112,112,112
130 DATA 66,0,156,3,0,4,0
140 DATA 5,0,6,0,7,0,8,0
150 DATA 9,0,10,0,11,0
160 DATA 12,0,13,0,14,0
170 DATA 15,0,2,0,2,16,2,32
180 DATA 2,48,2,64,2,80,2,96
190 DATA 2,112,2,65,0,6
```

If you have less than 48K then do not 'RUN' this program as it stands, slight modification will be required, this will be explained soon. Working our way through this program, we start with the first two lines which have a FOR/NEXT loop which READs 52 items of data (from 0 to 51), these values are then POKEd into locations 1536 onwards. Each value in the data statements represent the ANTIC value for the display list (see table 2). Now the display list resides in memory and all we need to do to enable it, is to point to it via locations 560 and 561. On line 30, locations 560 and 561 have been POKEd with 0 and 6 respectively.

Why 0 and 6 you may ask? The zero represents the LEAST SIGNIFICANT BYTE (LSB) and the 6 represents the MOST SIGNIFICANT BYTE (MSB), the LSB is always selected first by the computer and so is associated with the first location, 560, obviously the MSB is in location 561. To work out the value represented by the LSB and MSB, you have to realise that each byte can only be from 0 to 255, so to have larger numbers, for example 1578, the MSB is multiplied by 256 and the LSB is added to this value. For our example, to work out the MSB we have to find how many times 256 will go into 1578, ignoring any remainder. If you divide 1578 by 256 you get 6.164 etc., the only number we want is 6 which we can POKE into location 561 (MSB). To find the LSB we must know what the MSB (6) represents i.e. 6 lots of 256 which equals 1536, all that's left now is the difference between 1578 and 1536, this will always be between 0 and 255, in this case it is 42. So, in our program where we have stored the data at 1536 onwards the MSB is 1536 divided by 256 which equals 6 with no remainder, so the LSB must therefore be 0.

All the numbers in the data statements can be referenced to table 2 to see their function, except '66,0,156' in line 130 and '65,0,6' in line 190. We have already explained the 66 and the 65 earlier in this article, and we have also explained what the two following numbers represent.

In this display list the JVB instruction is followed by 0 and 6, which represent 1536, which makes the jump back to the beginning address (therefore, these two values will always be the same as the two values POKEd in locations 560 and 561). The LMS instruction (64) which has been added to a mode 2 line (equals 66) is followed by 0 and 156. These two point to a 'free' area of memory which we have arbitarily chosen. Referring back to the earlier point that if you have less than 48K, you must choose a value which will be in your memory limits. i.e. For 16K, 0 and 60 would be suitable. For 32K, 0 and 124 would be suitable values.

The rest of the program, lines 40 to 110, displays some information on the screen, in our case, the 'A's and the coloured pixels. Line 40 sets the variable 'S' to equal the start address of the display, i.e. 'S' must equal the actual address represented by the two numbers in the DATA statements following the LMS instruction (66). If the address was 1578 (MSB=6 and LSB=42), then line 40 would read: S=6*256+42 (or S=1578). Obviously, if you have changed the 0 and 156 in the DATA statements then line 40 must be changed accordingly, in our program the '+0' on line 40 has been omitted as 156*256=156*256+0.

Lines 50 and 60 are a 'FOR/NEXT' loop which places the code for an 'A' (33) into the first 200 bytes of the display ram, i.e. from 'S' to 'S'+199. These 200 bytes are the number of bytes needed by all 6 text modes, TABLE 3 shows the number of bytes required by each ANTIC display mode in all three widths, our program is in normal mode. The screen display is set to normal width at power-up and system-reset and can be changed by altering the value in location 559 (DMA control), this location is used for player-missiles and playfield width. Instead of giving all the possible combinations for this location, the easiest thing to say is if you want to get a narrow screen, subtract one from the contents of 559 or to get a wide screen, add one to the contents of 559. So, if you have a program using players on a normal width screen, somewhere in that program you will have POKEd 559 with a value, to get a narrow screen POKE it with one less, e.g. POKE 559,65 becomes POKE 599,64. If you are not using players the value in 559 will be 34 for a normal screen, therefore narrow and wide will be 33 and 35 respectively. As you can see how many bytes each line takes in normal mode, from TABLE 3, add up the numbers given for 2 to 7 and it will come to 200.

Lines 70 and 80 follow a similar function to 50 and 60 except that different information is displayed, the value of 27 which is stored displays the coloured blocks on the graphics modes from 8 to 15, note that the value of 27 could have been from 0 to 255, giving various combinations of blocks and colours for that byte. This loop covers the next 200 bytes which once again, if you add up the values given in TABLE 3 for modes 8 to 15, it will come to 200. Note that all the blank lines separating the display modes do not need any display ram, and therefore take 0 bytes and can be ignored. The rest of the display list, displays the blank lines and are separated by mode 2 lines, these of course do take display ram, and as there are 9 of them they take 9*40 bytes (360), and so the loop on lines 90 and 100 POKE zero's into all these lines. The zero represents a space character. To finish, line 110 loops back to itself to achieve a continuous display, until you press break or system-reset.

ANTIC MODE	NARROW SCREEN	NORMAL SCREEN	WIDE SCREEN
2	32	40	48
3	32	40	48
4	32	40	48
5	32	40	48
6	16	20	24
7	16	20	24
8	8	10	12
9	8	10	12
10	16	20	24
11	16	20	24
12	16	20	24
13	32	40	48
14	32	40	48
15	32	40	48

Table 3.

PRINTING TO A MODIFIED SCREEN

When you either modify a display list built by the computer or one of your own we recommend that you use POKE to access all parts of the screen, as PLOT, PRINT, etc. will not always produce the results you desire because the computer thinks you have a standard screen layout and if it is modified it may cause it to place data in a different position than that specified, due to different modes having different numbers of bytes per line.

Locations 88 and 89 are a two byte pointer and are used in a similar way to locations 560 and 561. The value represented by 88 and 89 tells the operating system (O.S.) where to find the start of the screen ram, therefore these two bytes are the same value as the two bytes following the LMS instruction in the display list. If you make 88 and 89 point to a position half-way down the screen then the O.S. will think that is the start of the display, and a PRINT or PLOT to the top-left position of that screen (0,0) will actually start from the half-way down position. This technique is useful when you have started in an O.S. mode and modified certain portions of that screen. To let the O.S. know which new mode you are PLOTing or PRINTing to, POKE the O.S. graphics number for that portion of the screen into location 87. For example if you started with a GRAPHICS 0 page and modified a strip to display some GRAPHICS 7 lines, then to PLOT to that area, POKE 87,7 to tell the O.S. that we wish to PLOT to the GRAPHICS 7 area. Now POKE 88 and 89 with the correct values to point to the start of this area and any subsequent PLOTs will treat the screen as starting from this strip. The changing of 88 and 89 is not essential as you can still PLOT to the GRAPHICS 7 area by doing a PLOT to within that area, leaving PLOT 0,0 in its original position. As you can well imagine this procedure is quite involved and it can be very complicated to achieve correct results. Unless you need to use the DRAWTO command then it is probably best to avoid the use of the screen editor. If you do need to use DRAWTO, then this will need alteration to any one or all of locations 87, 88 and 89, but this does not mean that you can not use direct POKEing to the screen ram.

To POKE direct to the screen ram all that you need to know is the start address of the screen ram, which, if you are using an O.S. mode then it will equal PEEK(88)+256*PEEK(89) or for a custom display it will be the two bytes following the LMS instruction. Now if you POKE to this value with a number, you will see it affects the top left position. Taking text modes first, each byte represents a single character, so if a text line is 40 bytes long it has 40 characters. Therefore, if you wish to print an 'A' somewhere in the middle of the screen, then to work out the address for the POKE, you multiply the number of lines down you wish to be by the number of characters per line and then add the number of columns in from the far left edge. e.g. To print 12 lines down and 20 columns in, if S equals the start of the screen then POKE S+(12*40)+20,33. The '33' represents the character 'A', but note this is not the same value as the ATASCII code. On page 55 of your BASIC reference manual you will find a chart showing the 'POKE' code for the characters. Unfortunately graphics modes are entirely different.

The graphics modes (modes 8 to 15) can be categorised into two types, 2 colour modes and 4 colour modes. Modes 9, 11, 12 and 15 are all 2 colour, and 8, 10, 13 and 14 are the 4 colour modes. These are in fact 1 and 3 colour modes because the background colour is included. For a two colour mode, each byte represents eight horizontally adjacent pixels, one bit for one pixel. A single bit can only be 1 or 0, which means that the zero represents the background colour (colour 0) and the one represents the other colour (colour 1). The background colour register for all these modes (except mode 15, which will be explained later) is location 712, colour 1 is location 708. Once you have selected a pattern of eight 1's and 0's for a byte, convert this binary number into a decimal number and POKE it into the appropriate memory location. Going back to colour registers, the contents of these can either be changed by a POKE or by a SETCOLOR command. First, decide which colour and luminance numbers you want for that register (which will be two numbers from 0 to 15) and multiply the colour number by 16 and add the luminance value, then POKE this value into one of the colour registers.

The four colour modes are very similar except that 2 bits are used for each pixel, giving only four pixels per byte. Each pair of bits will evaluate to a number from 0 to 3, representing the background colour and colours 1 to 3 respectively. The locations for colour registers 2 and 3 are 709 and 710 respectively, now when you work out the eight bits you must consider which of the four pixels you want on and then which colour register to use for them. For example 228 is 11100100, spliting this up into pairs we get 11 10 01 00. The first pair points to colour register 3, the second pair points to colour register 2 and the third pair points to colour register 1, the last being colour register 0 (background).

Mode 15 is different from the rest of the graphics modes because a zero in its bit pattern, which represents the background colour, actually resides in location 710 and a bit set to one represents a luminance of that colour, the luminance value is taken from location 709 (the colour value is ignored).

LOAD MEMORY SCAN

To finish off this article, we will describe the remaining instructions i.e. LMS and JMP, Vertical and Horizontal Scrolling, and Interrupts. We have already described that the LMS is used to point to the start address of the display data, but when using LMS's other factors should be taken into consideration.

There is a restriction in ANTIC that will not allow the 'memory scan counter' to be incremented over a 4K boundary. This means that you should start the display data area at a 4K boundary, i.e. multiples of 4K (4096), e.g. memory locations: 12288 (12K), 24576 (24K), 32768 (32K). By starting at a 4K boundary you can use the full 4K upto the next boundary for your data, of course if you started between boundaries the area available for your data is reduced, as you can only go upto the next boundary. If your data does cross a 4K boundary, then at the point on the screen where the data reaches the boundary, the data upto that point will be correct, but on reaching the boundary, the counter in ANTIC returns to the previous 4K boundary, thus causing the rest of your display data to be ignored, instead the data which is displayed after that point on the screen will be the data from the start of the previous 4K boundary.

Obviously, if the data for your screen is more than 4K long, you need a way of crossing a 4K boundary. The answer is to use a second LMS instruction to reload the counter to point to the next 4K block. If you start the display data at the beginning of a boundary then it is possible that the next boundary will occur somewhere in the middle of a line. If so, you cannot insert an LMS to reload the counter at that place, but at the start of the next line. Therefore you must move the start of the data, so that the 4K boundary occurs at the start of a new line, then you can insert the LMS on that line by adding 64 to the value. Do not forget that following this there are two extra bytes to be added, pointing to the 4K boundary, so that the display can carry on into the next 4K block. This is the technique used to create the operating system's GRAPHICS 8 page, as there are 7680 bytes of data for the screen. See listing 2 for a demonstration of boundary crossing.

Listing 2

```
10 REM Display boundary demo.
20 FOR I=0 TO 29
30 READ D:POKE 1536+I,D
40 NEXT I
50 POKE 560,0:POKE 561,6
60 REM Display start=11888
70 START=11888
80 REM 10 lines of 40 bytes=400.
90 REM boundary=12288=11888+400.
100 BOUNDARY=12288
110 REM Display 'A' upto boundary.
120 FOR I=0 TO 399
130 POKE START+I,33
140 NEXT I
150 REM Display 'B' after boundary.
160 FOR I=0 TO 399
170 POKE BOUNDARY+I,34
180 NEXT I
190 REM Display list data.
200 DATA 112,112,112,66,112,46
210 REM (112,46)=11888
220 DATA 2,2,2,2,2,2,2,2,2
230 DATA 66,0,48
240 REM (0,48)=12288
250 DATA 2,2,2,2,2,2,2,2,2
260 DATA 65,0,6
270 REM (0,6)=1536
```

Whilst on the subject of boundaries, we should point out that the display list itself must not cross a 1K boundary (1024). This is not really a problem as no display list could ever be over a thousand bytes long! The only reason your display list might have to cross a boundary is if there is no other room in memory, except over a 1K boundary (extremely unlikely!). If it does happen the JMP instruction with two following bytes can be implemented. The only problem with using this instruction, is that it automatically inserts a blank line on the display where the boundary occurs.

We have already shown the use of two LMS instructions to cross a boundary, but the use of more than one LMS in a display list is not just for crossing boundaries. In fact the LMS instruction can be used as often as you wish, to create many clever visual effects. The types of effect can be split into two main groups, static displays and dynamic displays e.g. scrolling. Both static and dynamic can be displayed simultaneously. Obviously, the limits are only bounded by your imagination, but here are some ideas. You could produce a 'mirror image' static display, i.e. imagine a mountain range mirrored in the waters of a lake. To produce this effect, you would need to define the top half of the screen to be the 'mountains' and use a single LMS to point to it, on the last line you would have an interrupt to change the 'character control' byte to invert the character set, and then for the lower half, you would have an LMS on every line to read the data backwards. i.e. The last line first, etc. Another use of LMS could be to point to some data which is stored separately from the main display data, so that, for example you could have a section of the screen with some permanent data displayed no matter how often the rest of the screen display changed.

SCROLLING

The dynamic form of LMS'ing is of course 'page flipping' and horizontal-vertical scrolling. Page flipping is the term given to switching the display between many set screens in memory very quickly, to give the effect of movement, a bit like flipping the pages of a book. To produce this effect only one LMS is needed, at the start of the display. By changing the two associated bytes to point to different blocks of data which have already been stored away, you in effect swap screen data. For example on a GRAPHICS 0 page the two bytes to be altered are the fifth and sixth numbers in the display list and if 'DL' pointed to the start of the display list then you should 'POKE' to DL+4 and DL+5 with the new values to point to your data.

Coarse scrolling can be achieved manually by drawing the required screen, erasing it and then redrawing it one 'block' further over or down. This is the way many micro's on the market achieve scrolling, but this is obviously a time consuming and very inefficient method. On the ATARI, this method could be used but the use of the LMS makes scrolling far easier and much more effective, and further to that fine scrolling is possible. Vertical scrolling is the easiest to implement as only one LMS instruction is necessary at the top of the display list. If you were to add the number of bytes for one line (see TABLE 3 for numbers of bytes) of the display to the two bytes of the LMS instruction, then the screen would effectively move up one line to the new start position, by repeating this process whole screens can be scrolled. Conversely, by subtracting the same amount the screen will scroll downwards. The point to notice is that the screen is like a viewing window which can be moved anywhere in memory to view the data therein. So, if you move the window down the data (e.g. from location 0 to 100), as the window is stationary the effect to the eye is that the screen is scrolling upwards.

Listing 3

```
10 REM Vertical coarse scrolling demo.
20 GRAPHICS 0
30 DL=PEEK(560)+256*PEEK(561)
40 REM Scroll the screen
50 REM from 0 to 2999 step 40 bytes.
60 FOR I=0  TO 2999 STEP 40
70 REM Work out the low and high bytes.
80 H=INT(I/256)
90 L=I-H*256
100 REM Move to that position.
110 POKE DL+4,L
120 POKE DL+5,H
130 REM Delay to slow down scroll.
140 FOR DEL=0 TO 100:NEXT DEL
150 NEXT I
160 GOTO 160
```

As you can see from the program above, all we have had to do to scroll the screen vertically, is to alter the low and high bytes and increment them by the number of bytes per line i.e. 40 bytes.

Horizontal scrolling needs multiple LMS's to move each line of the display. If you only wished to scroll one line you would still need at least two LMS instructions. The basic technique is to make the length of the line(s) to be scrolled longer than the screen display width. The idea is that you move the 'viewing window' over the length of the line. For example, if the screen width was 0 to 39 and you made the length of line from 0 to 127, then that line can be scrolled from 0 to 87, as the screen displays the last 40 bytes from 88 to 127. We chose 128 bytes per line because this is a factor of 4K, therefore if the screen goes passed a boundary, the following LMS will reload the counter anyway. In fact one line could be upto 4K long (but why you would want a line this long we don't know, as a 24 line mode 2 display would use up 96K of memory!!!!). Putting line lengths in factors of 4K (e.g. 64, 128, 256 etc.) is useful when designing a map, such as used in EASTERN FRONT, because there are no problems created when moving over any boundaries. To horizontally scroll a single line, the LMS two byte value should be incremented or decremented by an amount of one at a time.

Listing 4

```
10 REM Horizontal coarse scrolling.
20 FOR I=0  TO 24
30 READ D:POKE 1536+I,D
40 NEXT I
50 POKE 560,0:POKE 561,6:DL=1536
60 REM Start of scrolling line=12528
70 START=12528
80 FOR I=0  TO 127:POKE START+I,I:NEXT I
90 REM Scroll line.
100 FOR I=0  TO 88
110 REM Work out high and low bytes.
120 H=INT((START+I)/256)
130 L=(START+I)-H*256
140 REM Bytes 13 & 14 of the display list
150 REM point to the scrolling line.
160 POKE DL+12,L:POKE DL+13,H
170 FOR DEL=0 TO 30:NEXT DEL
180 NEXT I
190 GOTO 100
200 REM Data for display list.
210 DATA 112,112,112,66,0,48
220 DATA 2,2,2,2,2
230 DATA 66,240,48
240 DATA 66,112,49
250 DATA 2,2,2,2,2
260 DATA 65,0,6
```

As you can see from listing 4, the thirteenth and fourteenth bytes of the display list are being incremented by one from their initial value of 'START' (from line 100 to 180). Scrolling many lines, or even a whole screen at a time, involves having the LMS instruction, plus its two data bytes, on every line to be moved. Unfortunately, as BASIC is a 'slow' language, the scrolling of a whole screen will have a 'stepping' appearance running from top to bottom of the screen (or vice versa) because it is altering one LMS at a time, the more LMS's you have, the worse the effect (in BASIC). For example, if you were to have a whole screen scrolling, with 128 bytes per line, then the first LMS would point to the start of the screen data and each subsequent LMS would have a value of 128 bytes more than the last LMS i.e. a whole line more. This continues down the entire screen in multiples of 128, then to scroll the screen sideways by one character, just add or subtract one from every LMS in the display list, in the same manner as in our example listing for one line. If you have a whole screen scrolling horizontally, then to move the entire screen vertically, you must then add or subtract 128 bytes (or one line) to/from each LMS. The effect of vertically scrolling, followed by horizontally scrolling the screen is of course diagonal scrolling.

As mentioned earlier, it is possible to fine scroll on the ATARI, in fact it is very easy to implement. The idea is that fine scrolling is implemented during coarse scrolling, by two hardware registers named HSCROL and VSCROL, at addresses 54276 and 54277 respectively. If you change the contents of these registers, you will not see any scrolling at all on the screen as it is not yet enabled. The enabling of horizontal and vertical scrolling is achieved via the display list, by adding the value of 16, for horizontal, or 32 for vertical (see TABLE 2). Of course if you add both i.e. 48, then both are enabled together for that particular display line. So, if you wished to have a whole screen scrolling left/right and up/down, then you would add an LMS to each line and add the 48 to each LMS, to enable the fine scrolling. By changing the value in HSCROL from 0 to 3, you will see a mode 2 line scroll from left to right, leaving the line one 'shift' away from the next character block, i.e. in mode 2, four fine scroll movements equals one coarse scroll movement. Some modes will require this value to go upto 7 before the block is moved by the same amount. To get the line into the next block position, you do not increment by one more, but reset this value to 0, following this by a coarse scroll will give the effect of finely moving to the next block starting position. So in actuality, the screen is being finely scrolled, reset, coarse scrolled, finely scrolled, etc, thus producing a smoother movement to the eye. Vertical fine scrolling works on a similar basis, incrementing the value in VSCROL from 0 to 7 for mode 2 will cause that line to move upwards. By reseting and coarse scrolling the same effect is produced vertically. There is one important thing to notice, when vertical scrolling is enabled, one line of the display at the end of the vertically fine scrolled zone is in fact not displayed, thus producing a shorter screen. This should be accounted for by placing an extra display mode line outside the vertically scrolling zone. Add the following lines to listing 3 for a demonstration of fine scrolling.

```
21 VSCROL=54277
31 FOR I=0  TO 31
32 IF PEEK(DL+I)=66 THEN POKE DL+I,66+32
33 IF PEEK(DL+I)=2 THEN DL+I,2+32
34 NEXT I
121 POKE VSCROL,0
122 FOR V=0 TO 7
123 POKE VSCROL,V
```

In line 140, if you vary the delay you can see various degrees of smoothness. Add the following lines to listing 4, for a demonstration of fine horizontal scrolling.

```
11 HSCROL=54276
151 POKE HSCROL,3
161 FOR  H=3 TO 0 STEP -1
171 POKE HSCROL,H:NEXT H
```

In the data statements, change the 66 in line 230 to an 82, to enable horizontal scrolling. The delay in line 170 can also be changed to give varying speeds, we found 0 to 10 gave a reasonable rate. You will notice that even with fine scrolling, the display is still 'jerky'. To achieve a perfectly smooth scrolling movement, the scrolling must be synchronised with the T.V. scan rate. This is only possible by using machine code, but that would be another article!

DISPLAY LISTS

DISPLAY LIST INTERRUPTS.

The final instruction to be explained is the Display List Interrupt, nick-named 'DLI'. As its name suggests the interrupt is used to momentarily stop the main program, whether it is in BASIC or machine code, to implement an all-machine code interrupt routine, and once completed, returns to the main program, as if nothing had happened. The uses of DLI's are numerous and can be used for such things as: changing colours whilst drawing a screen, moving players, producing sounds, changing screen widths whilst drawing a screen, splitting a player into many segments, fine scrolling in different directions and at different speeds, e.g. Frogger.

To cause a DLI you must add 128 to the line in the display list, where you wish to cause the interrupt. 128 can be added anywhere including blank lines and 'jumps' and added as often as you wish. The interrupt can now be enabled and disabled by a single 'POKE' to NMIEN (54286), 64 disables and a 192 or higher will enable interrupts.
The two bytes at 512 and 513 should be altered to point to your interrupt routine, an RTI, machine code instruction will return operation to normal, at the end of your routine. You are also responsible for saving all registers on the hardware stack.

Listing 5 is a short program to display 256 colours at once, and includes the machine code interrupt routine in DATA statements, each one explained in the associated REM statement. Lines 160 and 170, point to and enable the machine code DLI routine.
This routine is from line 1190 to 1280, and from line 1290 to 1400 is a Vertical Blank routine to ensure that grey is always at the top of the screen. Once you have typed this program in and RUN it, you could use it like a T.V. test card to adjust the colours on your set to give the best picture.

Obviously, those of you who are not 'into' machine code will find the use of DLI's a bit beyond you, unfortunately, there is no easy way round this problem except learning machine code!

CONCLUSIONS

We hope that you have found this article interesting and informative. We have tried to explain the many intricate details of this subject, but when you get right down to it, there is no substitute for experimentation and practice.

Listing 5

```
10 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
20 REM xx 256 COLOURS...           xx
30 REM xx    By Keith Mayhew.      xx
40 REM xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
50 GRAPHICS 9:GOSUB 1000:GOSUB 1100
60 FOR C=1 TO 15
70 COLOR C
80 PLOT Cx5+4,191
90 DRAWTO Cx5+4,0
100 DRAWTO Cx5,0
110 POSITION Cx5,191
120 POKE 765,C
130 XIO 18,#6,0,0,"S:"
140 NEXT C
150 REM xxx ENABLE DLI INTERRUPTS xxx
160 POKE 512,0:POKE 513,6
170 POKE 54286,255
180 REM xxxxxxx ALL FINISHED xxxxxxxxxx
190 REM xxx JUST WAIT UNTIL STOPPED xxx
200 GOTO 200
1000 REM xxx CHANGE DLIST FOR DLI'S xxx
1010 DLIST=PEEK(560)+256xPEEK(561)
1020 FOR A=0 TO 6
1030 POKE DLIST+16+Ax12,143
1040 NEXT A
1050 FOR A=0 TO 7
1060 POKE DLIST+103+Ax12,143
1070 NEXT A
1080 POKE DLIST+3,207
1090 RETURN
1100 REM xxx READ IN MACHINE CODE xxx
1110 REM xxxxxxx TO PAGE SIX xxxxxxxx
1120 FOR X=0 TO 36
1130 READ DAT:POKE 1536+X,DAT
1140 NEXT X
1150 REM xxx ENABLE VERTICAL BLANK xxx
1160 RESULT=USR(1554)
1170 RETURN
1190 REM xxx DLI ROUTINE xxx
1195 REM PHA         SAVE A ON STACK
1200 DATA 72
1205 REM LDA $06FF   COLOR COUNT
1210 DATA 173,255,6
1215 REM STA $D40A   WAIT FOR SYNC
1220 DATA 141,10,212
1225 REM STA $D01A   CHANGE COLOR
1230 DATA 141,26,208
1235 REM CLC         SET CARRY
1240 DATA 24
1245 REM ADC #$10    ADD 16
1250 DATA 105,16
1255 REM STA $06FF   STORE BACK
1260 DATA 141,255,6
1265 REM PLA         RESTORE A
1270 DATA 104
1275 REM RTI         RETURN
1280 DATA 64
1290 REM xxxxx ENABLE ROUTINE xxxxx
1300 REM xxx ENTERED FROM BASIC xxx
1305 REM PLA        PULL A
1310 DATA 104
1315 REM LDA #$06   IMMEDIATE VBI
1320 DATA 169,6
1325 REM LDX #$06   HIGH POINTER
1330 DATA 162,6
1335 REM LDY #$1D   LOW POINTER
1340 DATA 160,29
1345 REM JSR $E45C  ENABLE
1350 DATA 32,92,228
1355 REM RTS        RETURN TO BASIC
1360 DATA 96
1370 REM xxx VERTICAL BLANK ROUTINE xxx
1375 REM LDA #$00   LOAD ZERO
1380 DATA 169,0
1385 REM STA $06FF  RESET COUNT
1390 DATA 141,255,6
1395 REM JMP $E45F  RETURN TO O.S.
1400 DATA 76,95,228
```

CONTACT

Listed below are members who are keen to meet or communicate with others, with a view to improving their knowledge of the ATARI computers, swaping hints, tips and programs, and generally getting to know other ATARI enthusiasts. So if you wish to 'Make Contact' with any of the people listed below, either because they are local to you, or you just wish to correspond by mail; write or phone (if the number is given) in the first instance. If you would like to have your name and address published, write to ROY at the club address with details.

LONDON
Tom Gainford, 15 Veryan Court, Park Road, London N.8.

KENT
Chris Davies, 71 Kingswood Avenue, Shortlands, Bromley, Kent.

SURREY
K. Ishaq, 39 Thirsk Road, Mitcham, Surrey, CR4 2BL. Telephone: 01-640 8607.

BUCKS
Daniel Lamb, 37 Chippendale Close, High Wycombe, Bucks, HP13 6BS.

CHESHIRE
Ian Skinner, 10 Sandhurst Road, Mile End, Stockport, Cheshire SK2 7NY.

DEVON
Ken Hall, 'Lyndhurst', Prospect, Okehampton, Devon. Telephone: 0837-2565.

TYNE & WEAR
Ian Scott, 70 Langholm Road, East Boldon, Tyne & Wear NE36 0EF.

OXON
Grahame Fairall, Shipton House, High Street, Shipton Wynchwood, Oxon, OX7 6BA.

NORFOLK
Norwich User Group, a club for ATARI owners, publishes a newsletter called NUGGET, contact Ken Ward for details: 45 Coleburn Road, Lakenham, Norwich, NR1 2NZ. Telephone: Norwich 661149.

U.S.A.
Seventeen-year-old American ATARI enthusiast wishes to correspond with similar age enthusiasts in England. Richard Repka, 132-46 60th Avenue, Flushing, N.Y. 11355, U.S.A.

USER GROUP SOFTWARE

Software Librarian Roy Smith

The basic rules of the library are as follows:

1. Every time you send to us a cassette or disk containing a program/ programs you are entitled to request three programs in return.
2. The program you send must be original and not copied.
3. There are no other rules (simple isn't it).

Ah! but life is never that simple, heres a few other things we need to point out. We can not process your contribution in the form of a print out, so cassettes or diskettes only. We recommend that with cassettes you record a back up copy on the 'B' side, if we cannot load your program we will return it for recording and resubmission. Also whilst on the subject of cassettes, we have had occasions where the programs requested will not fit onto the length of tape supplied, it is best to use a 'C60' minimum, a 'C90' would be even better. We also recommend using a high quality ferro-type tape such as TDK AD C60. When submitting on disk, sandwich it between two pieces of stiff card for protection.

A few points on the programs themselves: I like to use lines 0 to 9 for administrative purposes so start your program from line 10. If you have more than 16K, try to ensure that any player/ missile graphics or character set tables are placed in as low a place in memory as possible. Work out how much RAM is required to run the program.

Another thing to bear in mind is that you know how to use your program but other members who request a copy may not, so if any precise instructions are required for your program try to insert this information in the listing using REM statements.

Finally, the CLUB can not be held responsible for loss or damage to material whilst in the care of the GPO, also please include stamps to the value of 30p for return postage.

THE SOFTWARE LIBRARY SERVICE IS FOR MEMBERS ONLY

LIBRARY SOFTWARE TITLES

Listed below are the software titles available to members under the "three-for-one" exchange scheme. As can be seen, they are listed under basic program types, i.e. GAMES, UTILITIES, etc.; and also included is the minimum memory required. So if there is a title you fancy, just send in a program of yours for exchange.

Games

SHOOT
by Nigel Haslock – Switzerland.
Shoot down aircraft and helicopters, using your joystick controlled cannon! This program, written in BASIC, produces a machine code boot tape which runs with no cartridge. Novel feature is it can duplicate itself.
Runs in 16K Cassette minimum.

COLLISION COURSE
by Jon Beff – Manchester.
Avoid collisions with your opponent, but try to trap him into colliding with you. For two players or against the computer.
Runs in 16K Cassette or Disk minimum.

SUBMARINE HUNTER
by Hugh Denholm – Aberdeen.
Drop bombs from your helicopter to try and sink the sub, avoiding missiles from the protecting destroyer.
Runs in 16K Cassette or 32K Disk minimum.

SKYBLITZ VERSION 1
by Chris Barlow – Leigh.
Drop bombs from your spacecraft onto the buildings below, reduce them to rubble before you fly too low and crash into them.
Runs in 16K Cassette or Disk minimum.

SKYBLITZ VERSION 2
by Mike Barnard – Guisborough.
New version of Skyblitz 1, with improved graphics, sound and joystick control.
Runs in 16K Cassette or Disk minimum.

COWBOY
by Evan Fraser – Edinburgh.
Shoot your partner three times to kill him, reload with bullets from the local store.
Runs in 32K Cassette or Disk minimum.

BATTLE OF BRITAIN
by Mike Barnard – Guisborough.
A strategic game of wits, defending Britain against wave after wave of attacking bombers.
Runs in 16K Cassette or Disk minimum.

FRUIT MACHINE
by Mike Nash – Radstock.
A graphic representation of a fruit machine, where you can gamble the "computers money" Incorporates nudge and hold facilities.
Runs in 32K Cassette or Disk minimum.

YAHTZEE
by Steve Calkin – Basildon.
Dice game, where you have to get two, three or four of a kind, etc.
Runs in 16K Cassette or 32K Disk minimum.

MOONLANDER
by D. Mensing – Sutton Coldfield.
Manoeuvre your craft onto the landing pads.
Runs in 16K Cassette or 32K Disk minimum.

COUNTDOWN
by P. Stevens – Horley.
Hit moving targets with a bouncing ball and joystick controlled bats.
Runs in 16K Cassette or 32K Disk minimum.

HANGMAN
by R. L. Howarth – Preston.
Save the man from the gallows by guessing the word.
Runs in 16K Cassette or 32K Disk minimum.

MANIAC DRIVER
by P. J. Phillips – Sevenoaks.
Avoid oncoming traffic by skilful driving. This game uses PADDLES.
Runs in 16K Cassette or 32K Disk minimum.

PHANTOM FLAN FLINGER
by Chris Barlow – Leigh.
Throw flans into the dodging face and score points.
Runs in 16K Cassette or 32K Disk minimum.

COLOUR SNAP
by H. Clark – Barking.
ATARI version of the popular card game.
Runs in 16K Cassette or Disk minimum.

INVADERS
by Anthony Ball – Preston.
Although this version of space invaders is written in BASIC, it still retains the addictive quality of the original.
Runs in 48K Cassette or Disk (note this program will run in 16K Cassette or 32K Disk, but the titles may be corrupted).

FOUR IN A ROW
by R. P. Bosma – Canterbury.
Drop your marker in the grid and stop your opponent from getting four in a row.
Runs in 16K Cassette or Disk minimum.

DOGFIGHT
by Rod Knowles – Merseyside.
A game for two players involving World War I bi-planes in combat.
Runs in 32K Cassette or Disk minimum.

PEDESTRIAN
by P. Stevens – Horley.
You are the pedestrian and you must cross the road without getting run over.
Runs in 16K Cassette or 32K Disk minimum.

COLLISION COURSE 2
by Jon Beff – Manchester.
Improved version.
Runs in 16K Cassette or 32K Disk minimum.

LIBRARY SOFTWARE

ASTRO DODGER
by D. Dodson - Leigh-on-Sea.
Dodge between the asteroids to destroy the enemy craft.
Runs in 16K Cassette or 32K Disk minimum.

SKYBLITZ VERSION 3
by D. West - South Normanton.
Machine code version incorporating skill levels and faster action. This program is for use on DISK systems only. An added feature is that "HISCORES" can be written to disk. NB: source coding is available.
Runs in 32K Disk minimum.

REVERSI
by Ian Finlayson - Gosport.
BASIC version of this two player game with running scores.
Runs in 16K Cassette or Disk minimum.

CONNECT 4
by R. W. Askew - Northampton.
Use cunning and skill to stop your opponent from connecting four.
Runs in 16K Cassette or 32K Disk minimum.

GALACTIC CUBE
by Nigel Haslock - Switzerland.
Stear your craft to safety out of the space cube.
Runs in 16K Cassette or Disk minimum.

SKYBLITZ VERSION 4
by Jon Beff - Manchester.
Superb adaptation from the original, includes redefined character set to give enhanced graphic presentation.
Runs in 16K Cassette or 32K Disk minimum.

WORM
by Gordon Segar - Whitby.
Use your joystick to guide your worm through the tunnel without colliding with the jagged walls.
Runs in 16K Cassette or Disk minimum.

HANGMAN 2
by Carl Graham - Norwich.
Disk version of hangman, includes question files on cities and countries.
Runs in 48K Disk only.

FLIP
by Ezio Bottarelli - Italy.
Guessing game! Out guess the computer.
Runs in 16K Cassette or 32K Disk minimum.

JOIN FOUR
by Andrew Lusher - Erith.
Two player game, join four to win.
Runs in 16K Cassette or Disk minimum.

MOLE
by Keith Mayhew - Rayleigh.
Stop the moles from digging up your garden by hitting them on the head.
Runs in 16K Cassette or Disk minimum.

GOMOKU
by Ezio Bottarelli & Antonio Tocchi - Italy.
Two player game, get 5 counters in line to win.
Runs in 16K Cassette or Disk minimum.

FORZA QUATTRO
by Ezio Bottarelli & Antonio Sciarra - Italy.
Two player game, get four in a row.
Runs in 16K Cassette or Disk minimum.

TANKTRAP
by Ian Scott East Boldon.
Tank battle game for 1 or 2 players.
Runs in 16K Cassette or Disk minimum.

HEX
by Ezio Bottarelli & Antonio Tocchi - Italy.
Hexagonal game for two players, join your two sides before your opponent.
Runs in 16K Cassette or 32K Disk minimum.

ZAP
by Alex Macklen - Truro.
Move around the screen, eating up the energy pills, but be careful because your tail grows longer.
Runs in 16K Cassette or 32K Disk minimum.

SPACE FORTRESS
by Alex Macklen - Truro.
Protect your space fortress from the incoming missiles.
Runs in 32K Cassette or Disk minimum.

GUNFIGHT
by Ian Scott - East Boldon.
Two player gunfight; shoot to kill!
Runs in 32K Disk only.

FIFTEENS
by Ezio Bottarelli - Italy.
There are fifteen numbers that have got to be moved into the correct order.
Runs in 16K Cassette or Disk minimum.

PASSE DE TEMPS
by Desmond Seymour - Stoneleigh.
Get four barrels in the slots to win; 2 player game.
Runs in 32K Cassette or Disk minimum.

PECKMAN
by Mike Nash - Radstock.
Superb adaptation of a well known arcade game.
Runs in 16K Cassette or 32K Disk minimum.

TUNNEL TRILOGY
by Mark Christian - Wirral.
Guide your ship through the tunnel to the hidden city. Three games for the price of one (TUNNEL RUN, TUNNEL MASTER & TUNNEL'S REVENGE).
All run in 16K Cassette or 32K Disk minimum.

LANDER
by I. R. Skinner - Stockport.
Land your lunar module onto the landing pad.
Runs in 16K Cassette or 32K Disk minimum.

DEPTH CHARGE
by Ken Hall - Okehampton.
Drop depth charges onto the passing submarines.
Runs in 16K Cassette or 32K Disk minimum.

STUNT RIDER
by R. W. Askew - Northampton.
How many buses can you jump on your motorbike?
Runs in 16K Cassette or 32K Disk minimum.

Adventure Games

STONEVILLE MANOR
by Nigel Haslock - Switzerland.
Extensive BASIC word adventure, the object of which is to discover the treasures hidden in Stoneville Manor. Unfinished games can be saved onto cassette (slight modification of the program allows you to save to disk).
Runs in 32K Cassette or Disk minimum.

THE VALLEY
by Steve Calkin - Basildon.
Semi-graphic adventure, you can be warrior, wizard, priest, etc. and you must fight your way to safety along the forest path.
Runs in 32K Cassette or Disk minimum.

OUTPOST
by Anthony Ball - Preston.
Graphic adventure in which you defend the outpost from attacking enemies of varying strengths.
Runs in 32K Cassette or 48K Disk minimum.

THE FOLLY OF EZRHAR'D KKHANN!
by Alex Kells - Liverpool.
Journey through long dead EZRHAR'Ds dungeon looking for gold!
Runs in 48K Cassette or Disk minimum.

Home Entertainment

POLYGONS
by Chris Rutter - New Zealand.
Make polygons in Graphics 7 to 11, use your joystick to change the colours.
Runs in 16K Cassette or 32K Disk minimum.

LETTERFRAME
by Chris Davies — Bromley.
Sort out the letters into the correct order.
Runs in 16K Cassette or 32K Disk

SYNTHESISER
by Chris Payne - Manchester.
Program your keyboard to act as a synthesiser.
Runs in 16K Cassette or 32K Disk minimum.

DUNGEONS & DRAGONS CHARACTER GENERATOR
by A. J. Palmer - Basingstoke.
An absolute must for all dungeons and dragon players.
Runs in 16K Cassette or 32K Disk minimum.

BIORHYTHM
by Ezio Bottarelli - Italy.
Forecast your physical, intellectual and emotional future.
Runs in 16K Cassette or Disk minimum.

LIE DETECTOR
by D. Dodson - Leigh-on-Sea.
Instructions are included on how to make the hand held sensors. Then a display is given in the form of a graph showing true & false areas. As you hold the sensors and are asked awkward questions by a friend (?) he/she can see how often you LIE!!!
Runs in 16K Cassette or 32K Disk minimum.

DARTS SCOREBOARD
by Derek Harrison - Glasgow.
Let the computer keep score in your game of darts and give a fanfare to the winner.
Runs in 32K Cassette or Disk minimum.

NOUGHTS & CROSSES
by Ken Hall - Okehampton.
ATARI version of the popular game.
Runs in 16K Cassette or 32K Disk minimum.

DICE
by Carl Graham - Norwich.
Gamble on the roll of the dice!
Runs in 48K Disk only.

SEAWARFARE
by Steve Tullet - Dalkeith.
Computer version of that old favourite "Battleships".
Runs in 32K Cassette or Disk minimum.

LIBRARY SOFTWARE

PIG IN THE MIDDLE
by Keith Berry – Birmingham.
Card game, in which the two outside cards are displayed, and you can bet on the unrevealed middle card.
Runs in 16K Cassette or Disk minimum.

PICTURE PAINTER
by P. Patay – Oxted.
Paint pictures in GR.10, uses paddles as brushes.
Runs in 16K Cassette or 32K Disk minimum.

ROULETTE
by Carl Graham – Norwich.
Play the fascinating game of roulette and try to leave the casino with your shirt.
Runs in 16K Cassette or 32K Disk minimum.

SPIROGRAPH
by Andrew Lusher – Erith.
Draw patterns, dependant on user input.
Runs in 16K Cassette or 32K Disk minimum.

Demo's

SPIRAL
by Nigel Haslock – Switzerland.
Draw spirals of differing patterns depending on user input.
Runs in 16K Cassette or 32K Disk minimum.

ATARI CLOCK
by Ian Lawson-Smith – Watford.
An alarm clock for your home computer.
Runs in 16K Cassette or 32K Disk minimum.

ROCKETS
by Frank Silcock – Mountain Ash.
Demo showing rockets launching.
Runs in 16K Cassette or Disk minimum.

UNITED KINGDOM
by Stephen Salt – Lincoln.
3 option demo, map of U.K. Union Jack and National Anthem.
Runs in 32K Cassette or Disk minimum.

PLAYER MISSILE DEMO
by Keith Mayhew – Rayleigh.
A step by step demonstration of how to create player missiles on the ATARI.
Runs in 16K Cassette or Disk minimum.

256 COLOURS
by Keith Mayhew – Rayleigh.
A short program which will display all 256 colours available on the ATARI on the screen at once.
Runs in 16K Cassette or 32K Disk minimum.

COLOUR CORRIDOR
by Keith Mayhew – Rayleigh.
See the colours roll down the corridor.
Runs in 16K Cassette or 32K Disk minimum.

MEMORY SCROLLER
by Keith Mayhew – Rayleigh.
Scroll through memory a page at a time.
Runs in 16K Cassette or Disk minimum.

ATARI TRAIN
by Keith Berry – Birmingham.
Short demo incorporating player missile graphics.
Runs in 16K Cassette or Disk minimum.

SNOOPY
by Chris Davies – Bromley.
Sketches SNOOPY on the screen in graphics 8.
Runs in 16K Cassette or 32K Disk minimum.

SPHERES
by Peter Patay – Oxted.
Draws random spheres in graphics 9.
Runs in 16K Cassette or 32K Disk minimum.

QUADRANTS
by Peter Patay – Oxted.
A random pattern is generated in four positions to give kaleidoscopic effect.
Runs in 16K Cassette or Disk minimum.

PICASSO & PYTHAGORAS
by H. Clark – Barking.
Artistic patterns created by Pythagorian triangles.
Runs in 16K Cassette or Disk minimum.

PROBLEM & SOLUTION
by Ian Finlayson – Gosport.
A problem is set and a solution is given. Can you write a better program to solve the problem?
Runs in 16K Cassette or Disk minimum.

STERLING
by Allan Sharpe – Burgess Hill.
Character redefinition program, replaces "&" with a pound symbol.
Runs in 48K Cassette or Disk minimum.

U.S.S. ENTERPRISE
by Alex Kells – Liverpool.
Shows what can be achieved using simple Graphics 8 techniques.
Runs in 16K Cassette or 32K Disk minimum.

ART-6
by R. P. Bosma – Canterbury.
Six artistic demo's in Graphics 8.
Runs in 16K Cassette or Disk minimum.

TEAMUG
by Gordon Segar – Whitby.
Draws a mug of tea in graphics 10.
Runs in 16k Cassette or 32K Disk minimum.

RANDOM GEOMETRY
by Keith Berry – Birmingham.
Demo of randomly produced patterns incorporating squares, triangles and sound effects.
Runs in 16K Cassette or 32K Disk minimum.

MAINLAND G.B.
by J. Bennet – Newcastle.
Demo of different graphics modes (3-9 & 11) using map of Great Britain.
Runs in 16K Cassette or 32K Disk minimum.

Utilities

WEDGE
by Anthony Ball – Preston.
This program is for disk drive owners and it writes an AUTORUN.SYS file and adds commands to BASIC, such as KILL, RENAME, LOCK, UNLOCK, DIR, SCRATCH, without entering DOS.
Runs in 32K Disk minimum.

CURSOR FLASHER
by Jon Williams Littlehampton.
This is a machine code program which runs in vertical blank, and gives the following options:— FLASHING CURSOR, BLINKING CHARACTERS, INVERSE to NORMAL FLASHING, NORMAL to "SOLID WHITE", UPSIDE DOWN to NORMAL, and BLINKING INVERSE CHARACTERS. This program is also available in BASIC. When requesting this program please ask for either "CURSFLSH.BAS" or, if you want the source code, "CURSFLSH.BAS/SRC".
Runs in 16K Cassette or 32K Disk minimum.

FILER 1
by Chris Payne – Manchester.
A filing system for cassette owners.
Runs in 16K Cassette minimum.

ASSEMBLER
by Chris Rutter – New Zealand.
Create your own assembly language directly into memory. You can also save, move, list, modify and run programs from a menu.
Runs in 16K Cassette or Disk minimum.

OBJECT CODE TRANSLATOR
by Len Golding – Sheffield.
Assembly code which has been written using the ATARI ASSEMBLER EDITOR cartridge can be read and translated into DATA statements by this program, then re-written to disk or cassette for use in other programs. Please state if you require cassette or disk version of this program.
Runs in 16K Cassette or 32K Disk minimum.

CHARACTER GENERATOR 1
by Martin Walker – Swindon.
This program is for cassette owners, but could be adapted to disk. It allows you to modify all your 128 characters using the keyboard.
Runs in 16K Cassette minimum.

CHARACTER GENERATOR 2
by I. Scott – East Boldon.
This program allows you to modify up to 32 characters with joystick control giving such options as reverse, rotate, repeat and move. At the end it displays another program which allows you to use these new characters in any program you are writing.
Runs in 16K Cassette or 32K Disk minimum.

CASSETTE LOADER
by Jon Williams – Littlehampton.
Enables the user to load and save binary files to/from cassette. The load section of this program is compatable with object code produced by the ASSEMBLER EDITOR cartridge, so if you have trouble "CLOADing" from BASIC using ATARI ASSEMBLER EDITOR cartridge, this program is for you.
Runs in 16K Cassette minimum.

FAST SAVE CASSETTE
by Jon Williams – Littlehampton.
This program requires the use of the ATARI "ASSEMBLER EDITOR" cartridge and gives a faster way of saving binary programs.
Runs in 32K Cassette minimum.

GRAPHICS SHAPES
by Ken Ward – Norwich.
Re-defines character set to give circles, squares and other patterns but leaves standard letters and numbers intact.
Runs in 16K Cassette or Disk minimum.

CHARACTER GENERATOR 3
by J. Bennet – Newcastle.
Use joystick to draw new character in 8 by 8 grid. Press "C" to change to another character. Press "S" to stop, and obtain list of character and list of values for DATA Statement.
Runs in 16K Cassette or Disk minimum.

LIBRARY SOFTWARE

MORSE KEYBOARD
by Chris Barlow - Leigh.
Comprehensive Morse utility includes disk filing system for storing regularly used messages. Other features include speed and tone settings.
Runs in 16K Cassette or 32K Disk minimum.

DELETE
by Anthony Ball - Preston.
Gives microsoft delete function. This program is for disk owners.
Runs in 32K Disk minimum.

CREATOR
by Anthony Ball - Preston.
If you upgrade to a disk system from a cassette system, use this program to transfer data from cassette to autoboot disk.
Runs in 32K Disk minimum.

KEYBOARD
by Anthony Ball - Preston.
This program gives you faster repeat keys and shift/clear function. For use with disk systems.
Runs in 32K Disk minimum.

BIRTHDAY
by D. Dodson - Leigh-on-Sea.
For use by disk system owners to keep a file record of your family and friends birth dates.
Runs in 16K Disk minimum.

DISK FILE MANAGER
by D. Dodson - Leigh-on-Sea.
A disk file management system, so you can keep track of all your programs. The program is available with or without "PRINT" option, so state your requirement when asking for this program.
Runs in 48K or 32K Disk system minimum.

DISKCOPY
by Ken Hewitt - Nazeing.
Sector copy routine in BASIC, allows copying of disks which do not have DOS, but not protected software.
Runs in any size Disk system.

PROGRAM INDEX
by J. Bennet - Newcastle.
Cassette based program index, keeps up to 450 records.
Runs in 48K Cassette system only.

CHARACTER DESIGN AID
by Len Golding - Sheffield.
Allows you to redefine characters using a joystick, and then you can display the new character in 3 different graphics modes. Also you can design players and display them in 3 different sizes and 2 different resolutions.
Runs in 16K Cassette or 32K Disk minimum.

TOP TEN

1(5)	SYNTHESISER	CHRIS PAYNE
2(3)	THE VALLEY	STEVE CALKIN
3(1)	SHOOT	NIGEL HASLOCK
4(4)	ASSEMBLER	CHRIS RUTTER
5(6)	STONEVILLE MANOR	NIGEL HASLOCK
6(9)	MOONLANDER	D. MENSING
7(—)	LIE DETECTOR	D. DODSON
8(—)	ASTRO DODGER	D. DODSON
9(—)	OUTPOST	ANTHONY BALL
10(—)	BIORYTHM	EZIO BOTTARELLI

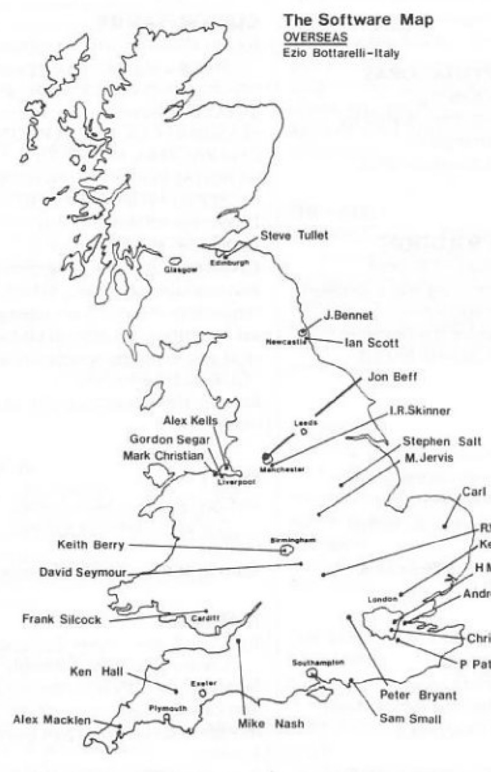
SHOOT has been knocked off the top spot this quarter and has been pushed down into third place! It looked at one stage as if VALLEY would make it to number one, but SYNTHESISER zoomed up on the rails from fifth position to clinch the premier position by a narrow margin, making it the most requested software exchange program since the last issue. Several new titles have moved into the chart, including two by Derek Dodson, who also wrote the reviews on joysticks in this issue.

SECTOR
by Ron Levy - Southend-on-Sea.
This program is a disk investigation aid.
Runs in 32K or 48K Disk systems.

FILEDUMP
by Peter Bryant - Maidenhead.
This program will PRINT any file that BASIC can read in either record or dump format.
Runs in 16K Cassette or Disk minimum.

DIRECTORY DISPLAY
by Ian Scott - East Boldon.
List diskette directory from BASIC.
Runs in any size Disk system.

CHECKSUM
by Ian Scott - East Boldon.
Run this program against a file containing a LISTed program to produce checksum data.
Runs in 16K Cassette or Disk minimum.



ADDRESS FILE
by J. Bennet - Newcastle.
Cassette address filing system.
Runs in 16K minimum Cassette only.

COMPUTER ASSISTED DESIGN
by Sam Small - Bognor Regis.
Design and draw different circular shapes and view them at varying angles.
Runs in 32K Cassette or Disk minimum.

CATALOG
by H. M. Hoffman - London.
Catalogue system.
Runs in 32K Cassette or Disk minimum.

PLAYER DESIGNER
by Keith Berry - Birmingham.
Design players with this useful program.
Runs in 16K Cassette or Disk minimum.

Education

MORSE TUTOR
by Chris Barlow - Leigh.
This program generates random morse at selected speeds so you can teach yourself morse code.
Runs in 16K Cassette or 32K Disk minimum.

KEYBOARD TUTOR
by Mike Jervis - Nottingham.
Learn to touch type with this typing tutor.
Runs in 16K Cassette or Disk minimum.

MATH TEST
by Mike Jervis - Nottingham.
Fun with figures for your children.
Runs in 16K Cassette or Disk minimum.

Music

MUSIC 1
by Graham Ward - Liverpool.
A selection of 4 tunes for use with the ATARI MUSIC COMPOSER cartridge. The tunes are: THE QUEEN OF SHEBA by Handel, MINUET 1 by Bach, MINUET 2 by Purcell and BRITISH GRENADIERS.
Runs by 16K Cassette or 32K Disk minimum.

MUSIC 2
by H. M. Hoffman - London.
A Selection of 5 tunes for use with the ATARI MUSIC COMPOSER cartridge, the tunes are: YESTERDAY, YELLOW SUBMARINE, SCARBOROUGH FAIR, AULD LANG SYNE and THE NEW WORLD SYMPHONY (DVORAK).
Runs in 16K Cassette or Disk minimum.

INTERFACE

Atari Service

Dear Sirs,

Thank you for issue three, it was worth waiting for. After having a few problems with my 410 program recorder, I sent it back to ATARI and approximately eight weeks later it was returned, I thought perhaps they were re-building it piece by piece. As you can imagine, not being able to use my computer for all that time, I got down to some serious work. The cassette player worked fine for about a month and then started to play up again. I cleaned the heads with a proper head cleaner and it worked for three or four times, but then I would be back to square one. I tried using better quality tapes (TDK type) but to no avail, I usually use C15 'computer' tapes.

A frienc' said that my problems might be caused by magnetic particles on the recording heads due to fast forwarding and rewinding on cheap tapes, could this be so? You can buy a special tape to demagnetize the heads, would it be worth purchasing this, as I know you must be careful not to damage the heads(?) I bought my ATARI 400 last Christmas, and I have already lost four months work because of this recorder and I am most reluctant to send it back to ATARI again, having waited so long last time.

Thank you for your article 'BYTEING DEEPER INTO BASIC', it has answered a lot of questions that I would like to have asked, although I still do not understand the GOSUB command.

If I saw a short program with a GOSUB routine and a similar program without, to see the difference, I would understand it better. It is a pity that in the BASIC manuals they do not have short programs with explanations by the side of each line, telling you what that command or statement does. I have written some small programs (recipes with graphics for my wife) and have tried the GOSUB command to try and shorten the program, but have come unstuck. I did put the POP command in though and it worked, how I don't know. Like Mr. Ward, I am at the age where the ultimate question of life, the universe and everything does not really interest me, but trying to understand BASIC and graphics on my ATARI does, although it is an uphill struggle. I am going to get the DE RE ATARI manual, but I was wondering if there are any other books that explain BASIC's operation more fully?

My 400 is a 16K model, I get on fine with typing things in on the flat keyboard, in fact I am quite quick. I want to upgrade to 32K or 48K so that I can buy a disk drive (that will solve my problems with my 410). You used to be able to get back a refund on your old 16K card but now as the 800 is supplied with 48K standard this practice has stopped.

Do you think it would be better to sell my 400 and purchase an 800 or upgrade my 400 to 48K, which costs about 90 pounds? Does the keyboard really matter as long as you can use it OK or are the proper keys superior and less likely to go wrong?

Is the keyboard the only difference between the 400 and the 800?

I am also thinking of buying a secondhand colour portable TV, unless the computer will work on a balck and white set or would it do harm to my computer?

I typed in many of the programs in issue three and was very pleased with the quality, especially PEDESTRIAN. I think Mr. Stevens is a very clear man, I enjoyed playing his game tremendously.

I have typed in programs from glossy magazines and most of them did not work, although I know hardly anything about programming, I really think that they come nowhere near the professionalism of the programs you printed in issue three. Please keep up the good work, because I feel sure that everybody who receives your newsletters gets pure enjoyment out of reading, learning and typing in the excellent programs.

E. A. Symons — Burgess Hill, Sussex.

COMMENT

Sorry to hear that it took ATARI so long to repair and return your 410 recorder, we can only assume all this happened before the service centres were set up, as we believe the service is far better organised now and consequently should not take more than one or two weeks. It is a popular misconception that the short 'computer tapes' are more reliable.

In fact most are simply cheaper tapes cut to a shorter running time. From our experience we would recommend TDK AD C-60 tapes, others may be suitable. As to the point on 'magnetic particles' deposited on the tape head, this will only occur on play or record since this is the only time the tape is touching the head.

Generally speaking the cheaper tapes do leave more particles. Cleaning of the heads will help this problem, use a cotton bud dipped in methylated spirits. You should also run the edge of your nail slowly and firmly up and down the metal drive spindle. Purchasing a tape head demagnetiser is not really necessary. To help you with your problems with 'GOSUB', we would suggest that you disregard the 'POP' command until you have a full understanding of the 'GOSUB' command, which we will not cover here. Some books which you may find useful on such problems are: 'ATARI BASIC (a self teaching guide)' by ALBRECHT, FINKEL and BROWN; 'ATARI BASIC LEARNING BY USING' by THOMAS E. ROWLEY; 'BASIC EXERCISES FOR THE ATARI' by J. P. LAMOITIER, all of which are available from MAPLIN.

As you mentioned De Re ATARI we recommend that you do not purchase this manual yet as it is of a more advanced nature.

About changing your machine to an 800, if you are quite content with the flat keyboard there are virtually no advantages except you can use a video monitor and external hi-fi sound output, and of course there is an extra cartridge slot but at present it is not put to any use. Upgrading your 400 to 48K would seem to be a better option, and it only costs sixty-five pounds from MAPLIN, but they do keep the 16K card. Your computer will work with a black and white TV, or in fact no TV at all! Of course many of the games will lose their appeal without colour.

Novice

Dear Chris,

I am a novice in the computer field and I have a few queries that I hope you can sort out for me. First, I have a 48K 400 which seems to be faulty, let me explain. . . After switching on, if I "?FRE(0)" I only get about 37K, and after loading up DOS I get even less. Should this happen or is there something wrong? I must point out though that I have never had any trouble with programs.

The other point is that if I leave the computer switched on and go away for a while, when I come back the screen has changed colour and seems to flick through varying colours after every few seconds. Once I have touched the keyboard it goes back to normal.

From C. Grimes — Luton.

COMMENT

The figures you are getting after "?FRE(0)" are correct, the reason for this is that any cartridge that you have plugged in, presumably the BASIC cartridge in this case, overshadows an area of RAM, thus giving the lower reading. The same thing applies when you load up DOS.

As to the colour changes, this too is not a fault. The ATARI has an automatic screen protection system which is called the "ATTRACT MODE". If after 9 minutes a key has not been pressed, the computer starts a continuous rotation of colours to the screen to prevent "burn through" which could happen if a colour is displayed for too long.

If you were to peek into location 77 you would see a gradual increase in numbers, starting from 0 up to 127, when suddenly number 254 is displayed. Number 254 implements the ATTRACT MODE and the screen starts to change colour. Thus it is not hard to figure out that if location 77 always contained a number less than 127 (or 127) the ATTRACT MODE would not be triggered. This could be useful to know if you are writing a program of a game, say, which may well run for a fairly long period. Of course, by using POKE 77,254 the ATTRACT MODE could be implemented during a program to give the opposite effect.

The Valley

Dear Sirs,

I must congratulate you on the ever improving quality of the UK Atari Computer Owners' Club magazine. The latest format was a pleasure to read and had a lot of useful information and programs in it.

I was very pleased to see my 'Valley' conversion program at No. 3 in your charts, and it is this program that has prompted me to write.

Some while ago, I was contacted by a young lady who was a member (I think) of a Chelmsford user group. She had exchanged a program for 'The Valley' and phoned me to see if there were any instructions as she was having difficulty playing it. Similarly, Ezio Bottarelli in Italy contacted me and we now have a good friendship developing, which is nice.

I have sent instructions for 'The Valley' to Ezio, but could not do so for this mysterious young lady as I do not know her name and address. I did say that if she sent me a large SAE, I would despatch the instructions, but this never arrived.

Could you please check your file and make sure that you have got the instructions that I sent in with the program, and if not, I will send a duplicate set. I would also appreciate it if you could either print this letter or make it known in your next issue that anybody who has 'The Valley' without instructions can contact me directly.

Thanks again for a good support magazine. A fellow Atari-ite.

Steve Calkin — Pitsea

COMMENT
We have looked through our files and we do not have a copy of instructions to send out with 'VALLEY'. We will obtain a copy from you, for future members use. If members who already have VALLEY would like to write to MR CALKIN his address is: 10 THE POPLARS, PITSEA, BASILDON, ESSEX.

Assembler Problems

Dear Sirs,

Congratulations on a great magazine, I find it really interesting and useful. Please could you answer a few questions. Recently I bought a book called 'PROGRAMMING THE 6502' by Rodnay Zaks. In the book it has a list of 6502 instructions with their binary equivalents. Unfortunately some instructions have a binary equivalent composed of '1's, '0's and 'b's. i.e. LDY = 101bbb00 and JMP= 01b01100.

What do the 'b's mean? Also, I have some programs in assembly language but not an ASSEMBLER/EDITOR cartridge, so I would like to be able to 'POKE' these listings into memory. I understand that if I see a 'JSR' I must POKE a 32 into whatever location I am at and if I see, for example, alpha (which has been previously defined as say $CB then I must POKE in 203. In issue two of the newsletter you have an assembler routine to move players vertically, with the program on the next page to load it into memory. Just for practice I started to work out the PLA's and INY's into decimal and checked them on the next page. But I had a problem at line 290 which reads: 0290 BCC UP. I wasn't sure what to do about the 'UP' since it hadn't been previously defined, the same with line 300 where it reads: 0300 BNE DN.

Do I ignore the 'UP' and 'DN' or what? Also at line 370 you have 'LDA(FROM),Y'. The 'Y' seems to have been ignored in the decimal listings, is that because of the comma preceeding it? At line 460 you have: 0460 LOOP1 DEY. Would you have to try and convert the LOOP1 or just ignore it? Elsewhere in issue two there is a cursor flasher program and at 560 it reads: 0560 INIT PLA. Would you ignore the 'INIT' or is it vital to the program? On line 580 it reads: 0580 LDX #ENTRY/256 SET VVBLKI. I understand that a '#' should load the register with whatever follows it but how can it load: 'ENTRY/256 SET VVBLKI'?

I would be very grateful if you could answer these questions. Also do you know if the new ATARI 1200 computer will be out soon and what new features will it have?

Paul Barber — Kings Lynn, Norfolk.

COMMENT
You have made some fundamental errors in your assumptions about assembly language in general. Unfortunately it is beyond the scope of this column to explain these to you. We would recommend anyone who is considering learning about machine code, that the purchase of the ATARI assembler editor cartridge is absolutely essential.
All the problems you have mentioned are in fact explained in 'PROGRAMMING THE 6502' by Rodnay Zaks. The new range of ATARI computers are shown in several magazines. Several of the machines, including the 1200 will not be sold in this country, but the rest of the range will be sold here.

Dogfight Mods.

Dear Sirs,

The UK Atari Computer Owners Club magazine is just ace! I enjoy all the articles and listings.

The reason for writing is because I am concerned that some 16K computer owners may be losing out. For example, I (or my wife!) typed in all the listings which were labelled 16k but baulked at the idea of further hours spent typing in those labelled 32K. However, we are so hooked on Atari that we will be purchasing one of the new 64k models when they become available later in the year, and in preparation for this happy event, I typed in two 32K programmes on my Atari 400 to save them on cassette for future use. To my delight one of these, "Fruit Machine" by Mike Nash (issue 2) worked as listed in 16K! Further, "Dogfight" by Rod Knowles worked in 16K with only slight alterations. Dogfight is such a delightful game that I am appending my modifications which (a) fit it into 16K and (b) improve the building and cloud layout (as far as 16K allows), so others may include the game in their collections.

(a) Delete lines 32, 34, 36, 4360
Delete all lines with REM statements
Line 10 GR.17:POSITION 6,22:?#6; "DOGFIGHT": GOSUB 4000
Line 550 Alter position location to POSITION 16,22
Line 650 Alter position location to POSITION 3,22
Line 5010 Alter end of line to POSITION 3,22:?#6; "0"
Line 5020 Alter start of line to POSITION 16,22:?#6; "0"
Line 6004 Take out POSITION 4,23:?#6; "game over":
Line 6020 POSITION 3,22:?#6; "0": POSITION 16,22:?#6; "0": GOTO 20
Delete line 5000 and renumber line 5005 as 5000

(b) Line 4520 NWCH=NWCH+8: FOR X=NWCH TO NWCH+15
Line 4535 FOR X=NWCH+48 TO NWCH+87: READ C: POKE X,C: NEXT C
Delete lines 4560, 4570, 4580, 4590
Line 5020 POSITION 16,22:?#6; "0"
Line 5030 COLOR 34: PLOT 5,20: DRAWTO 14,20: DRAWTO 6,19: DRAWTO 6,11: DRAWTO 7,19: DRAWTO 13,19: DRAWTO 13,11: DRAWTO 12,19
Line 5050 COLOR 136: PLOT 3,2: PLOT 5,5: PLOT 7,3: PLOT 12,1: PLOT 14,6: COLOR 137: PLOT 4,2: PLOT 6,5: PLOT 8,3: PLOT 13,1: PLOT 15,6
Line 5055 COLOR 138: PLOT 3,3: PLOT 5,6: PLOT 7,4: PLOT 12,2: PLOT 14,7: COLOR 139: PLOT 4,3: PLOT 6,6: PLOT 8,4: PLOT 13,2: PLOT 15,7
Note you need a POKE 82,0 to type in the last two lines. Also insert a line 2 POKE 82,0 to do this when the programme is run.

Steve Tullet — Dalkeith

COMMENT
It's strange about FRUIT MACHINE! This is not the first time we have been told that it should work in 16K, even Mike Nash has spoken to us about it. But we can only say everytime we have tried to run it in 16K it will not go! But we will try it again and try to solve the mystery once and for all.
Thank you for your modifications to DOGFIGHT, we have printed them above and we hope many members will find further enjoyment from this two player game.

All members' comments should be addressed to CHRIS BARLOW, P.O. Box 3, Rayleigh, Essex. Please include S.A.E. for a reply.

SPECIAL OFFER

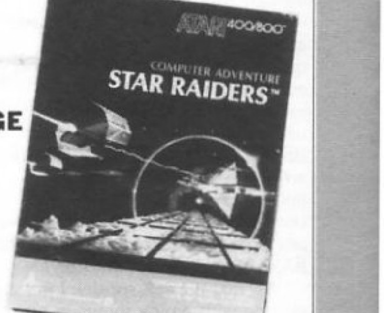
This is a genuine offer, and all the software are originals. Because of the tremendous response from the last issue, we have persuaded MAPLIN to offer our members a "Star Offer" this time. The highly acclaimed ATARI cartridge "STAR RAIDERS" is available at a whole £10 off the usual price. This game is a true 3-dimensional space chase game, with many levels, from novice to expert. When many of the other games have faded, you'll come back to this one time and time again.

Take full advantage of these special offers, by filling in the order coupon on the insert, or write your requirements on a piece of paper and send it, with a cheque made payable to M.E.S. Ltd, to: CHRIS BARLOW, THE U.K. ATARI COMPUTER OWNERS CLUB, P.O. BOX 3, RAYLEIGH, ESSEX.

STAR OFFER!

STAR RAIDERS CARTRIDGE

Usual price £29.95,
special offer price £19.95.
Order code: YG66W



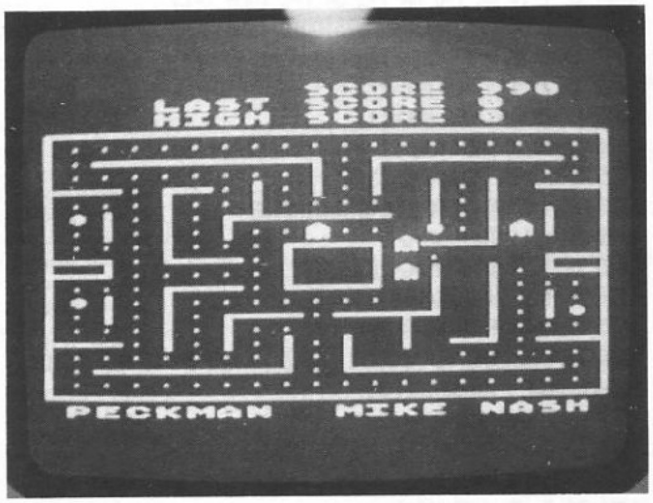
Runs in 16K Cassette or 32K Disk

PECKMAN

by Mike Nash – Radstock

As the name suggests, this game is similar to a well known arcade game; we'll leave you to guess which one! Although written in BASIC, the overall effect of the game is very good. In fact we think it's very, very good; we loved it. When you first RUN the game, the screen goes blank for a few seconds and then a title page is displayed. Next the maze is drawn, and very quickly too! Then it's up to you to peck your way around the maze, eating as many dots as you can, whilst being chased by the "GREEN MEANIES". If you eat the large dots, the meanies turn "BLUE" with fear, and rightly so, because now you can eat them. Be extra careful in this game, as you only have one life; but if you do clear the maze, there is a nice little bonus routine.

NOTE: In this program, anything which is underlined, should be entered in "INVERSE".



```
15 DATA P,108,e,96,C,91,k,81,M,72,a,64,n,60
16 DIM A$(1)
18 GRAPHICS 18:POSITION 3,5:SOUND 0,217,10,4:FOR I=1 TO
7:READ A$,B:? #6;A$;" ";:FOR J=9 TO 1 STEP -1
20 SOUND 1,B,10,J:NEXT J:FOR K=1 TO 150:NEXT K:NEXT I:SO
UND 0,0,0,0:SOUND 1,0,0,0:FOR I=1 TO 900:NEXT I
22 FOR I=0 TO 10:POSITION 0,I:? #6;"xxxxxxxxxxxxxxxxxxxx
":FOR K=1 TO 50:NEXT K:NEXT I
45 SC=0:HSC=0:LSC=0:DIM GOLD(4),GPOS(4)
50 POKE 756,224:GOSUB 28000
55 L=PEEK(88)+PEEK(89)*256
58 SETCOLOR 0,9,8:SETCOLOR 1,1,12:SETCOLOR 2,2,8:SETCOLO
R 3,12,10
60 FOR I=0 TO 479:READ Z:POKE L+I,Z:SOUND 0,Z,8,6:NEXT I
:SOUND 0,0,0,0
61 GOSUB 1500
62 FOR I=1 TO SCR+1:FOR N=9 TO 1 STEP -0.5:SOUND 0,I*10,
10,N:NEXT N:NEXT I:RESTORE 30101
65 OLDC=71:OLDY=15:Y=15:G=349:DOT=185:GL1=0:MV=0:MV1=20:
MV2=1:GFLAG=0:G2FLAG=0:K=0:GC=205:BN=0
75 FOR I=1 TO 4:GOLD(I)=0:NEXT I:GPOS(1)=249:GPOS(2)=250
:GPOS(3)=269:GPOS(4)=270:YY=21
95 POSITION 15,1:? #6;"     ":POSITION 15,1:? #6;LSC:POS
ITION 15,2:? #6;HSC
100 POSITION 15,0:? #6;SC
105 YY=YY+1:IF YY=15 THEN FOR I=1 TO 4:POKE L+GPOS(I),20
5:NEXT I:MV1=-MV1:MV2=-MV2:GC=205
106 IF YY=15 THEN SOUND 3,0,0,0
110 I=1:GOSUB 1200:SOUND 0,50,2,8
112 IF K=1 THEN 150
115 I=2:GOSUB 1000:SOUND 0,0,0,0:GOSUB 1200:IF K=1 THEN
150
120 I=3:GOSUB 1200
122 IF K=1 THEN 150
124 GOSUB 1000
125 I=4:GOSUB 1200
127 IF GFLAG=99 THEN I=2:GOSUB 1200
130 IF K=1 THEN 150
132 IF DOT<=0 THEN 175
135 GOTO 100
150 LSC=SC:IF SC>HSC THEN HSC=SC:SC=0
151 SC=0:SCR=0
155 RESTORE 30000:SETCOLOR 1,12,10:POSITION 0,23:? #6;"P
RESS START TO PLAY"
160 FOR J=0 TO 250 STEP 5:SOUND 1,J,10,12:FOR X=1 TO 5:N
EXT X:NEXT J:SOUND 1,0,0,0:SOUND 0,0,0,0
162 FOR X=0 TO 100:NEXT X:SOUND 1,70,4,10:FOR X=1 TO 150
:NEXT X:SOUND 1,0,0,0
164 X=0
165 IF PEEK(53279)=6 THEN 170
166 X=X+1:IF X<3000 THEN 165
167 FOR X=0 TO 459:POKE L+X,0:NEXT X
168 POSITION 4,10:? #6;"- PECKMAN -"
169 GOSUB 200
170 GOTO 58
175 REM SCREEN CLEARED
180 POSITION 0,23:? #6;"CONGRATULATIONS     "
181 SOUND 3,0,0,0:SCR=SCR+1
185 RESTORE 30111:GOSUB 800:POSITION 0,0:? #6;CHR$(125):
IF BN=0 THEN 190
186 POSITION 6,3:? #6;"bonus":POSITION 0,12:FOR I=1 TO B
N:? #6;"-";:FOR K=90 TO 1 STEP -1:SOUND 0,I*5,10,K
187 NEXT K:NEXT I:BN=BN*100*SCR:SC=SC+BN:SOUND 0,0,0,0
188 POSITION 6,18:? #6;"X";SCR*100:FOR I=1 TO 500:NEXT I
189 ? #6;CHR$(125):FOR K=0 TO 19:POSITION 0,K:? #6;"bon
us ";BN;" bonus ";BN:NEXT K:FOR I=1 TO 500:NEXT I
190 RESTORE 30000:GOTO 60
200 I=220
205 POKE L+I,75:FOR A=1 TO 100:NEXT A:POKE L+I,0:POKE L+
I+1,71:FOR A=1 TO 100:NEXT A:POKE L+I+1,0
208 POKE 77,0
210 IF PEEK(53279)=6 THEN RETURN
211 IF INT(RND(0)*10)>1 THEN 215
212 Z=L+I+1:POKE Z,74:FOR A=1 TO 100:NEXT A:POKE Z,0:POK
E Z-20,75:FOR A=1 TO 100:NEXT A:POKE Z-20,73
213 FOR A=1 TO 100:NEXT A:POKE Z-20,0
215 I=I+2:IF I<240 THEN 205
220 GOTO 200
800 REM TUNE
805 READ P1:READ T1:IF P1=99 THEN SOUND 0,0,0,0:RETURN
810 T1=T1*20:FOR J=1 TO T1:SOUND 0,P1,10,8:NEXT J
820 SOUND 0,0,0,0:GOTO 805
1000 REM MOVE GOBBLER
```

```
1002 IF Y<>15 THEN OLDY=Y
1003 OLDC=NWC:Y=STICK(0)
1004 IF Y=10 OR Y=6 OR Y=5 OR Y=9 THEN Y=15
1005 IF OLDC<>75 THEN NWC=75:GOTO 1030
1007 XX=Y:IF Y=15 THEN XX=OLDY
1010 IF XX=11 THEN NWC=72
1015 IF XX=14 THEN NWC=74
1020 IF XX=7 THEN NWC=71
1025 IF XX=13 THEN NWC=73
1030 IF Y=15 THEN POKE L+G,NWC:RETURN
1040 IF Y=11 THEN NG=G-1
1045 IF Y=14 THEN NG=G-20
1050 IF Y=7 THEN NG=G+1
1055 IF Y=13 THEN NG=G+20
1060 Z=PEEK(L+NG):IF Z<>140 AND Z<>0 AND Z<>13 AND Z<>13
9 THEN POKE L+G,NWC:RETURN
1065 POKE L+G,0:G=NG
1070 IF Z=140 THEN SC=SC+10:DOT=DOT-1
1075 IF Z=13 THEN 1105
1080 IF Z=139 THEN 1090
1085 POKE L+G,NWC:RETURN
1090 REM SPOT EATEN
1095 FOR X=1 TO 4:POKE L+GPOS(X),13:NEXT X:SOUND 3,250,2
,6
1100 GSC=200:GC=13:MV1=-MV1:MV2=-MV2:YY=0:GOTO 1085
1105 REM GHOST EATEN
1106 SC=SC+GSC:POSITION 0,0:? #6;GSC:GSC=GSC*2:BN=BN+1
1110 FOR X=1 TO 4:IF G=GPOS(X) THEN A=X
1111 NEXT X
1112 IF GOLD(A)=140 THEN DOT=DOT-1
1115 GPOS(A)=209:GOLD(A)=PEEK(L+209):IF PEEK(L+209)=GC T
HEN GPOS(A)=210:GOLD(A)=PEEK(210+L)
1120 FOR X=0 TO 250 STEP 5:SOUND 2,X,10,10:NEXT X
1125 FOR Y=1 TO 4:FOR X=250 TO 0 STEP -10:SOUND 2,X,10,1
0:NEXT X:NEXT Y:SOUND 2,0,0,0
1127 POSITION 0,0:? #6;"     "
1130 GOTO 1085
1200 REM GKOST1
1210 IF GFLAG=99 OR I>2 THEN 1235
1215 READ GFLAG:IF GFLAG=99 THEN 1235
1220 X=L+GPOS(I):POKE X,GOLD(I):GPOS(I)=GPOS(I)+GFLAG:X=
X+GFLAG:GOLD(I)=PEEK(X)
1225 IF GOLD(I)>70 AND GOLD(I)<76 AND GC=205 THEN K=1:RE
TURN
1230 POKE X,GC:GOTO 1430
1235 IF G2FLAG=99 OR I<3 THEN 1260
1236 IF GFLAG<>99 THEN FOR X=1 TO 50:NEXT X:GOTO 1430
1240 READ G2FLAG:IF G2FLAG=99 THEN 1260
1245 X=L+GPOS(I):POKE X,GOLD(I):GPOS(I)=GPOS(I)+G2FLAG:X
=X+G2FLAG:GOLD(I)=PEEK(X)
1250 IF GOLD(I)>70 AND GOLD(I)<76 AND GC=205 THEN K=1:RE
TURN
1255 POKE X,GC:GOTO 1430
1260 R1=INT(GPOS(I)/20):C1=GPOS(I)-(R1*20)
1265 R2=INT(G/20):C2=G-(R2*20)
1270 IF R1<R2 THEN MV=MV1
1275 IF R1>R2 THEN MV=-MV1
1277 IF R1=R2 THEN X=INT(RND(0)*2):IF X=0 THEN X=-1
1280 IF R1=R2 THEN MVA=MV*X:GOTO 1300
1285 X=PEEK(L+MV+GPOS(I))
1290 MVA=-MV
1295 IF X>70 AND X<76 AND GC=205 THEN K=1:RETURN
1297 IF X<>140 AND X<>0 AND X<>34 THEN 1300
1298 GOTO 1400
1300 IF C1<=C2 THEN MV=MV2
1305 IF C1>C2 THEN MV=-MV2
1310 X=PEEK(L+MV+GPOS(I))
1315 IF X>70 AND X<76 AND GC=205 THEN K=1:RETURN
1320 IF X<>140 AND X<>0 AND X<>34 THEN MV=-MV:GOTO 1350
1325 GOTO 1400
1350 X=INT(RND(0)*3):IF X>1 THEN MV=MVA
1352 X=PEEK(L+MV+GPOS(I))
1355 IF X>70 AND X<76 AND GC=205 THEN K=1:RETURN
1360 IF X<>140 AND X<>0 AND X<>34 THEN 1430
1400 SOUND 1,GPOS(I),10,6:Z=L+GPOS(I):POKE Z,GOLD(I):Z=Z
+MV:GPOS(I)=GPOS(I)+MV:SOUND 1,0,0,0
1405 GOLD(I)=PEEK(Z):POKE Z,GC
1430 REM
1450 RETURN
1500 REM REMOVE SPOTS
1505 IF SCR>0 THEN POKE L+181,140:DOT=DOT+1
1510 IF SCR>1 THEN POKE L+198,140:DOT=DOT+1
1520 IF SCR>2 THEN POKE L+301,140:DOT=DOT+1
1525 POKE 77,0
1550 RETURN
28000 REM CHAR SET ROUTINE
28007 DIM B$(80)
28010 POKE 756,224
28040 MEMEND=PEEK(106)-4:CHSET=MEMEND*256
28060 POKE 106,MEMEND-1:GRAPHICS 1+16
28070 FOR X=1 TO 32
28075 READ A:B$(X,X)=CHR$(A)
28080 NEXT X
28082 DATA 104,104,133,213,104,133,212,104,133,215,104,1
33,214,162,4,160,0,177,212,145,214
28083 DATA 200,208,249,230,213,230,215,202,208,240,96
28090 I=USR(ADR(B$),224*256,CHSET)
28105 REM MODIFY 4 CHARS
28110 FOR X=1 TO 18
28120 READ CHAR:N=CHSET+CHAR*8:FOR I=0 TO 7:READ T:POKE
N+I,T:NEXT I
28140 NEXT X
28600 DATA 13,24,60,126,90,126,126,126,90
28610 DATA 6,0,12,30,30,30,30,12,0
28620 DATA 7,0,12,30,60,56,60,30,12
28630 DATA 8,0,24,60,30,14,30,60,24
28635 DATA 9,0,0,8,28,62,62,54,34
28640 DATA 10,0,0,34,54,62,62,28,8
28645 DATA 11,0,24,60,60,60,24,0,0
28650 DATA 12,0,0,0,16,16,0,0,0
28651 DATA 14,24,24,24,31,31,0,0,0
28652 DATA 15,0,0,0,31,31,24,24,24
28653 DATA 26,0,0,0,248,248,24,24,24
28654 DATA 27,24,24,24,248,248,0,0,0
28655 DATA 28,0,0,0,255,255,0,0,0
28656 DATA 29,24,24,24,24,24,24,24,24
28657 DATA 30,24,24,24,255,255,0,0,0
28658 DATA 31,0,0,0,255,255,24,24,24
28659 DATA 32,24,24,24,31,31,24,24,24
28660 DATA 59,24,24,24,248,248,24,24,24
```

continued on 23

COMPUTER ASSISTED DESIGN

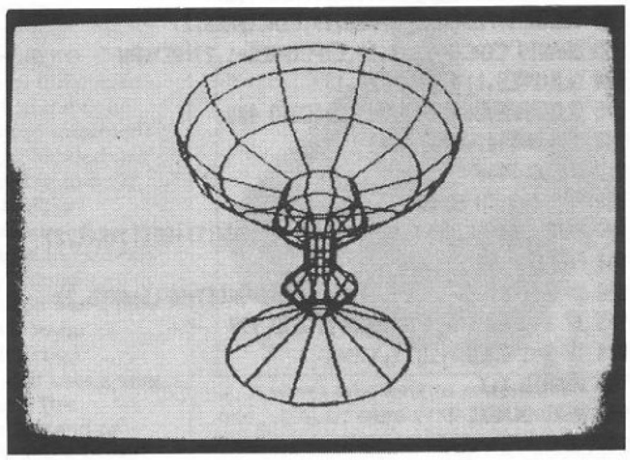
by Sam Small — Bognor Regis

This program generates a three dimensional, cylindrical model from an initial two dimensional template. Various options are provided from a menu, to give different viewing angles and more or less construction lines. All designs are displayed on the highest resolution graphics mode to provide accurate detail.

NOTE: In this program, anything which is underlined, should be entered in "INVERSE".

Runs in 32K Cassette or Disk

```
2 DIM Q$(1):GOTO 1200
3 DIM K(33,2)
5 DIM CIRC(30,2),OLD(30,2)
6 DIM COLOR$(1),RF(30),VW(30)
7 DIM Q$(1):PI=22/7:ARC=(PI/2)/90
8 GOSUB 9500:GOSUB 9200:RETURN
12 GOSUB 7000
13 C=(RND(0)*14)+1
14 SETCOLOR 2,C,0
15 SETCOLOR 4,C,0
16 SETCOLOR 1,C,8
30 ? CHR$(125)
40 ? "______________________________"
45 ? " C O M P U T E R   A S S I S T E D "
46 ? "______________________________"
47 ? "           D E S I G N           "
50 ? "______________________________"
52 RETURN
100 C=RND(0)*16
110 GRAPHICS 8+16
120 SETCOLOR 1,C,0
130 SETCOLOR 2,C,10
140 SETCOLOR 4,C,10
145 COLOR 1
150 RETURN
160 GOSUB 100
200 FOR L=1 TO HORS
210 DIS=(150/(HORS-1))
220 CIRC(L,1)=180
230 CIRC(L,2)=((DIS*L)-DIS)+21
235 OLD(L,1)=CIRC(L,1)
236 OLD(L,2)=CIRC(L,2)
240 NEXT L
250 RETURN
300 FOR D=1 TO HORS
320 PLOT CIRC(D,1)-40,CIRC(D,2)
330 NEXT D
335 PLOT 160,0:DRAWTO 160,191
336 GOSUB 600
340 PLOT CIRC(1,1),CIRC(1,2)
360 FOR D=2 TO HORS
370 DRAWTO CIRC(D,1),CIRC(D,2)
380 NEXT D
400 IF STRIG(0)=0 THEN GOSUB 600
410 S=STICK(0)
415 IF PEEK(53279)=6 AND VERT>1 AND ANG>0 THEN GOSUB 7000
:GOTO 900
416 IF PEEK(53279)=6 AND ANG<1 THEN 1200
417 IF PEEK(53279)=6 AND VERT<1 THEN GRAPHICS 0:GOTO 1200
419 IF PEEK(53279)=5 THEN GRAPHICS 0:GOTO 1200
```



```
420 IF S=15 THEN 400
430 IF S=14 THEN CIRC(PRES,2)=CIRC(PRES,2)-1:GOTO 800
440 IF S=13 THEN CIRC(PRES,2)=CIRC(PRES,2)+1:GOTO 800
450 IF S=11 THEN CIRC(PRES,1)=CIRC(PRES,1)-1:GOTO 800
460 IF S=7 THEN CIRC(PRES,1)=CIRC(PRES,1)+1:GOTO 800
470 IF S=6 THEN CIRC(PRES,1)=CIRC(PRES,1)+1:CIRC(PRES,2)=
CIRC(PRES,2)-1:GOTO 800
475 IF S=5 THEN CIRC(PRES,1)=CIRC(PRES,1)+1:CIRC(PRES,2)=
CIRC(PRES,2)+1:GOTO 800
480 IF S=9 THEN CIRC(PRES,1)=CIRC(PRES,1)-1:CIRC(PRES,2)=
CIRC(PRES,2)+1:GOTO 800
485 IF S=10 THEN CIRC(PRES,1)=CIRC(PRES,1)-1:CIRC(PRES,2)
=CIRC(PRES,2)-1:GOTO 800
499 GOTO 400
600 PRES=PRES+1
610 IF PRES=HORS+1 THEN GOTO 690
620 RUB=PRES-1
630 IF RUB=0 THEN RUB=HORS
640 COLOR 2
650 PLOT 130,0:DRAWTO 130,191
660 COLOR 1
670 PLOT 130,CIRC(PRES,2)
685 RETURN
690 COLOR 2:PLOT 130,0:DRAWTO 130,191:PRES=1
691 PLOT 140,0:DRAWTO 140,191:COLOR 1
695 FOR T=1 TO HORS
696 PLOT 140,CIRC(T,2):NEXT T:GOTO 620
800 IF CIRC(PRES,2)<=0 THEN GOSUB 9000:CIRC(PRES,2)=1:GOT
O 400
805 IF CIRC(PRES,2)>=191 THEN GOSUB 9000:CIRC(PRES,2)=190
:GOTO 400
810 IF CIRC(PRES,1)<=160 THEN GOSUB 9000:CIRC(PRES,1)=161
:GOTO 400
820 IF CIRC(PRES,1)>=319 THEN GOSUB 9000:CIRC(PRES,1)=319
:GOTO 400
825 IF PRES=1 THEN GOSUB 850:GOTO 890
830 IF PRES=HORS THEN GOSUB 840:GOTO 890
835 GOSUB 840:GOSUB 850:GOTO 890
840 COLOR 2:PLOT OLD(PRES,1),OLD(PRES,2)
841 DRAWTO OLD(PRES-1,1),OLD(PRES-1,2)
844 COLOR 1:PLOT CIRC(PRES,1),CIRC(PRES,2)
```

```
845 DRAWTO CIRC(PRES-1,1),CIRC(PRES-1,2):RETURN
850 COLOR 2:PLOT OLD(PRES,1),OLD(PRES,2)
851 DRAWTO OLD(PRES+1,1),OLD(PRES+1,2)
854 COLOR 1:PLOT CIRC(PRES,1),CIRC(PRES,2)
855 DRAWTO CIRC(PRES+1,1),CIRC(PRES+1,2):RETURN
890 OLD(PRES,1)=CIRC(PRES,1)
895 OLD(PRES,2)=CIRC(PRES,2):GOTO 400
900 GOSUB 100:GOSUB 4000
910 GOSUB 3000
920 FOR T=1 TO HORS
930 PLOT 160+(CIRC(T,1)-160)*K(1,1),VH(T)+RF(T)*K(1,2)
940 FOR D=1 TO 33
950 X=160+(CIRC(T,1)-160)*K(D,1):Y=VH(T)+RF(T)*K(D,2)
965 IF Y<0 OR Y>191 THEN B=1:GOTO 999
970 IF B=1 THEN PLOT X,Y:B=0
980 DRAWTO X,Y
999 NEXT D:NEXT T
1000 C=(2*PI)/VERT
1005 FOR X=0 TO (2*PI)-C STEP C
1006 SX=SIN(X):CX=COS(X)
1010 FOR J=1 TO HORS-1
1020 RD1A=CIRC(J,1)-160
1030 RD1B=CIRC(J+1,1)-160
1080 X1=160+RD1A*CX:Y1=VH(J)+RF(J)*SX
1090 X2=160+RD1B*CX:Y2=VH(J+1)+RF(J+1)*SX
1100 IF Y1<0 THEN Y1=0
1110 IF Y1>191 THEN Y1=191
1120 IF Y2<0 THEN Y2=0
1130 IF Y2>191 THEN Y2=191
1140 PLOT X1,Y1
1150 DRAWTO X2,Y2
1180 NEXT J:NEXT X
1185 GOSUB 5000
1190 IF STRIG(0)=1 THEN 1190
1200 GRAPHICS 0:GOSUB 12
1205 ? :? :? "    Construction lines        UP    "
1210 ? :? "    Change veiwing angle     DOWN  "
1220 ? :? "    Go to drawing page       LEFT  "
1230 ? :? "    Begin new design         RIGHT "
1240 ? :? "    Draw current design      FIRE  "
1242 ? :? "    Help                     OPTION "
1245 ? :? :? "        USE JOYSTICK TO SELECT      ";
1250 S=STICK(0)
1255 IF PEEK(53279)=3 THEN 10000
1260 IF S<>15 THEN 1300
1270 IF STRIG(0)=0 THEN 1440
1280 GOTO 1250
1300 IF S=14 AND HORS>1 THEN GOSUB 12:GOTO 1350
1310 IF S=13 AND HORS>1 THEN GOSUB 12:GOTO 1360
1320 IF STICK(0)=11 AND HORS>1 THEN GOSUB 100:PRES=HORS:G
OSUB 660:GOSUB 7000:GOTO 335
1330 IF S=7 THEN 1400
1340 GOTO 1250
1350 ? :? :? :? "      Number of lines in current"
1354 ? :? "      design is ";VERT;"."
1356 ? :? :? :? "       Change to ";:INPUT VERT:GOTO 1200
1358 GOTO 1200
1360 ? :? :? :? "      Angle of veiw in current"
1362 ? :? "      design is ";ANG;" degrees."
1364 ? :? :? :? "       Change to ";
```

```
1366 INPUT ANG:GOSUB 3000:GOTO 1200
1400 GOSUB 12:? :? :? :? "  Do you want a new design (Y/N
)";:INPUT Q$:IF Q$="N" THEN 1200
1410 CLR :GOSUB 3:GOSUB 12
1420 ? :? :? "  How many construction circles";:INPUT HOR
S:CIRCLES=1:GOSUB 200
1430 GOTO 1200
1440 IF HORS<1 THEN 1250
1445 IF VERT<1 THEN GOSUB 12:GOTO 1350
1450 IF ANG<1 THEN GOSUB 12:GOTO 1360
1460 IF HORS>1 THEN GOSUB 7000:GOTO 900
1470 GOTO 1250
3000 FOR V=1 TO HORS
3005 A=90-ANG
3010 IF CIRC(V,2)<96 THEN H=96-CIRC(V,2):N=INT(H*SIN(ARC*
A)):VH(V)=CIRC(V,2)+(H-N):GOTO 3100
3020 H=CIRC(V,2)-96:N=INT(H*SIN(ARC*A)):VH(V)=CIRC(V,2)-(
H-N)
3100 NEXT V
3110 RETURN
4000 FOR V=1 TO HORS
4005 G=CIRC(V,1)-160
4010 RF(V)=INT(G*SIN(ARC*ANG))
4020 NEXT V
4030 RETURN
5000 SOUND 0,10,10,14
5010 FOR X=1 TO 7
5020 FOR T=1 TO 3:NEXT T
5030 SOUND 1,20,10,14
5040 FOR T=1 TO 3:NEXT T
5050 SOUND 1,30,10,14
5060 NEXT X
5070 SOUND 0,0,0,0
5075 SOUND 1,0,0,0
5080 RETURN
7000 SOUND 0,10,10,14
7010 SOUND 1,15,10,14
7020 FOR X=1 TO 8:NEXT X
7997 SOUND 0,0,0,0
7998 SOUND 1,0,0,0
7999 RETURN
8000 IF CIRC(PRES,1)>319 THEN CIRC(PRES,1)=319:GOSUB 9000
:RETURN
8010 IF CIRC(PRES,2)<0 THEN CIRC(PRES,2)=1:GOSUB 9000:RET
URN
8020 IF CIRC(PRES,2)>191 THEN CIRC(PRES,2)=191:GOSUB 9000
:RETURN
9000 SOUND 0,200,10,14
9010 SOUND 1,200,6,14
9020 FOR X=1 TO 8:NEXT X
9030 SOUND 0,0,0,0
9040 SOUND 1,0,0,0
9050 RETURN
9200 FOR D=1 TO 33
9205 X=X+0.2
9210 K(D,1)=COS(X):K(D,2)=SIN(X)
9220 NEXT D:RETURN
9500 GOSUB 12:X=0
9510 POSITION 6,10:? "(C) SAM SMALL  AUG 1983 (V.2)"
9520 POSITION 8,16:? " Please wait a moment ";
```

COMPUTER ASSISTED DESIGN

```
9530 RETURN
10000 GOSUB 12
10010 ? :? "[1] CLOAD and RUN."
10020 ? "[2] MENU displayed."
10030 ? "[3] All functions are locked out          excep
t 'new design'."
10040 ? "[4] Select 'new design'."
10050 ? "[5] 'New design Y/N' is displayed for      check
ing."
10060 ? "[6] Program creates COS and SIN arrays     for f
aster circle drawing."
10070 ? "[7] Select how many construction           circl
es are required."
10080 ? "[8] Return to MENU."
10090 ? "[9] Drawing page is now available."
10100 ? :? :? "      PRESS 'OPTION' TO CONTINUE";
10105 IF PEEK(53279)<>3 THEN 10105
10110 GOSUB 12
10120 ? :? "[10] DRAWING PAGE"
10130 ? :? "     Select which segment to be drawn      wi
th fire button."
10140 ? :? "     Joystick will then 'rubber band'      dr
awing line to the desired"
10150 ? "     profile. All eight directions are      suppo
rted and there are limit          checks."
```

```
10160 ? :? "     START then draws design if             ot
her parameters have been            entered."
10170 ? :? "     SELECT goes back to MENU"
10180 ? :? "      PRESS 'OPTION' TO CONTINUE";
10190 IF PEEK(53279)<>3 THEN 10190
10200 GOSUB 12
10210 ? :? "[11] MAIN MENU"
10220 ? :? "     Attempting to 'draw current            de
sign' without entering               other parameters ";
10230 ? "will result in "
10240 ? "     program asking for these entries."
10250 ? :? "     'Draw current design' will             dr
aw three-dimentional                 representation of ";
10260 ? "drawn profile."
10270 ? :? :? :? "      PRESS 'OPTION' TO CONTINUE";
10275 IF PEEK(53279)<>3 THEN 10275
10280 GOSUB 12
10290 ? :? :? :? "[12] PARAMETERS"
10300 ? :? "     'Construction lines'                   An
y number from 1 upwards."
10310 ? :? "     'Angle of veiw'                         0
deg - eye level,                     90 deg - overhead."
10320 POSITION 4,20:? "  PRESS 'OPTION' FOR MAIN MENU";
10340 IF PEEK(53279)<>3 THEN 10340
10350 GOTO 1200
```

Continued from 20

PECKMAN

```
28700 POKE MEMEND-1,0:POKE 756,MEMEND
28710 RETURN
30000 REM MAZE
30001 DATA 0,0,0,0,0,0,0,0,0,51,35,47,50,37,0,0,0,0,0,0
30002 DATA 0,0,0,0,44,33,51,52,0,51,35,47,50,37,0,0,0,0,
0,0
30003 DATA 0,0,0,0,40,41,39,40,0,51,35,47,50,37,0,0,0,0,
0,0
30004 DATA 15,28,28,28,28,28,28,28,28,28,28,28,28,28,28,
28,28,28,28,26
30005 DATA 29,140,140,140,140,140,140,140,140,140,140,14
0,140,140,140,140,140,140,140,29
30006 DATA 29,140,28,28,28,28,28,28,28,26,140,15,28,28,2
8,28,28,28,140,29
30007 DATA 29,140,140,140,140,140,140,140,140,29,140,29,
140,140,140,140,140,140,140,29
30008 DATA 32,28,28,140,15,28,140,29,140,29,140,29,140,2
9,140,29,140,28,28,59
30009 DATA 29,140,140,140,29,140,140,29,140,140,140,140,
140,29,140,29,140,140,140,29
30010 DATA 29,139,29,140,29,140,15,30,28,28,28,28,140,29
,140,29,140,29,139,29
30011 DATA 29,140,29,140,29,140,29,140,140,140,140,140,1
40,140,140,29,140,29,140,29
30012 DATA 29,140,140,140,29,140,140,140,15,28,28,26,140
,28,28,27,140,140,140,29
30013 DATA 32,28,26,140,14,28,28,140,29,205,205,29,140,1
40,140,140,140,15,28,59
```

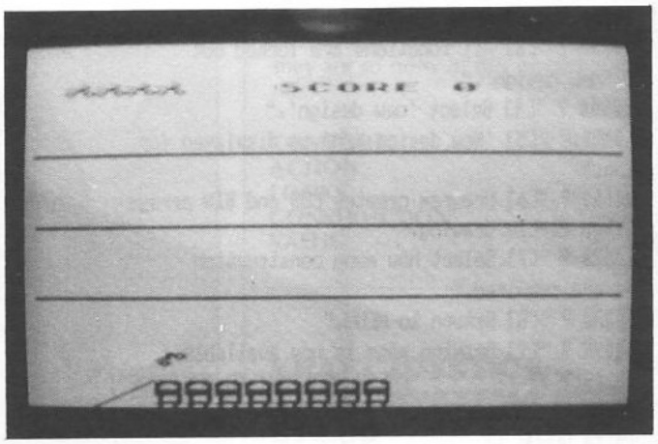
```
30014 DATA 32,28,27,140,140,140,140,140,29,205,205,29,14
0,29,140,29,140,14,28,59
30015 DATA 29,140,140,140,15,28,28,140,14,28,28,27,140,2
9,140,29,140,140,140,29
30016 DATA 29,139,29,140,29,140,140,140,140,140,140,140,
140,29,140,29,140,29,140,29
30017 DATA 29,140,29,140,29,140,15,28,28,140,28,28,31,27
,140,29,140,29,139,29
30018 DATA 29,140,140,140,29,140,29,140,140,71,140,140,2
9,140,140,29,140,140,140,29
30019 DATA 32,28,28,140,29,140,29,140,29,140,29,140,29,1
40,28,27,140,28,28,59
30020 DATA 29,140,140,140,140,140,140,140,29,140,29,140,
140,140,140,140,140,140,140,29
30021 DATA 29,140,28,28,28,28,28,28,27,140,14,28,28,28,2
8,28,28,28,140,29
30022 DATA 29,140,140,140,140,140,140,140,140,140,140,14
0,140,140,140,140,140,140,140,29
30023 DATA 14,28,28,28,28,28,28,28,28,28,28,28,28,28,28,
28,28,28,28,27
30024 DATA 0,176,165,163,171,173,161,174,0,0,45,41,43,37
,0,46,33,51,40,0
30050 DATA 162,1,128,1,121,1,108,4,162,1,128,1,121,1,108
,4,162,1,128,1,121,1,108,2,128,2,162,2,128,2,144,4
30051 DATA 99,99
30100 REM GHOST MOVES
30101 DATA -40,-40,-1,1,-1,1,20,20,20,20,99
30102 DATA -60,-60,-1,1,-1,1,99
30110 REM TUNE 2
30111 DATA 121,1,108,1,96,1,91,3,96,4,99,99
```

STUNT RIDER

Runs in 16K Cassette or 32K Disk

by R. W. Askew — Northampton

Test your skill at jumping over parked buses. You use your joystick to control the speed of your motorbike as it hurtles towards the ramp. You have to judge your speed just right or, even though you jump all the buses, you will still crash if you are going too fast. The number of buses you have to jump will vary on every run so there is no chance to judge from your mistake the time before.



NOTE: In this program, anything which is underlined, should be entered in "INVERSE".

```
1 ? "INITIALIZING"
10 CB=PEEK(742)*256-1024
14 FOR A=0 TO 511:POKE CB+A,PEEK(57344+A):NEXT A:FOR A=0
 TO 79:READ B:POKE CB+A,B:NEXT A
15 GRAPHICS 17:POKE 756,CB/256:POKE 708,132:POKE 709,52:
POKE 710,0:POKE 711,196
20 POKE 712,200:POSITION 4,4:? #6;"stunt rider":POSITION
 8,6:? #6;"BY"
25 POSITION 3,8:? #6;"bob  and nick":POSITION 7,10:? #6;
"askew"
30 ST=(PEEK(106)-16)*256:POKE 54279,ST/256:FOR A=0 TO 25
6:POKE ST+1024+A,0:NEXT A:Y=150
35 FOR A=0 TO 7:READ B:POKE ST+1024+Y+A,B:NEXT A
40 POKE 53277,3:POKE 704,0:SOUND 0,130,2,2:POKE 559,62
50 FOR X=25 TO 220:POKE 53248,X:NEXT X:POSITION 4,15:? #
6;"HOLD  START"
60 IF PEEK(53279)=6 THEN SOUND 0,0,0,0:POSITION 4,15:? #
6;"           ":GOTO 850
61 GOTO 50
74 ? #6;CHR$(125):POKE 559,0
76 EB=1:U=5:C=0:POKE 712,136
80 POKE 77,0:BUS=INT(RND(0)*11)+6
81 POSITION 8,0:? #6;"SCORE ";C
85 Q=165:J=0:EX=0
100 COLOR 1:PLOT 4,21:DRAWTO BUS,21:COLOR 2:PLOT 4,22:DR
AWTO BUS,22
105 IF U>1 THEN COLOR 165:PLOT 1,0:DRAWTO U-1,0
110 COLOR 163:PLOT 2,22:PLOT 3,21
120 COLOR 132:PLOT 0,15:DRAWTO 19,15:PLOT 0,10:DRAWTO 19
,10:PLOT 0,5:DRAWTO 19,5
190 FOR A=0 TO 7:POKE ST+1024+Y+A,0:NEXT A:X=50:Y=64:RES
TORE 1090:FOR A=0 TO 7:READ B:POKE ST+1024+Y+A,B:NEXT A
192 POKE 54286,192:POKE 559,62:POKE 53248,X:SOUND 0,200,
2,2
195 IF STRIG(0)=0 THEN P=1:GOTO 200
196 GOTO 195
200 X=X+P:IF X>220 THEN X=40:GOSUB 250
210 SOUND 0,180-INT(P)*7,2,2
212 POKE 53248,X
215 IF STICK(0)=14 THEN P=P+0.025*EX:EX=EX+0.125
216 IF STICK(0)=13 AND P>1 THEN P=P-0.1
240 GOTO 200
250 IF Y>110 THEN POP :POKE 53248,0:M=0:N=22:GOTO 300
255 RESTORE 1090:FOR A=0 TO 7:POKE ST+1024+Y+A,0:READ B:
POKE ST+1024+Y+40+A,B:NEXT A:Y=Y+40
260 RETURN
300 K=INT(P)*2
305 IF K<3 THEN K=4
306 IF K>17 THEN K=17
308 Z=70-(K*4)
309 COLOR Q:PLOT M,N:IF M>0 THEN COLOR 0:PLOT M-1,N
310 FOR A=1 TO Z:NEXT A
320 M=M+1
330 LOCATE M,N,L:IF L=163 THEN Q=166:N=N-1:COLOR 0:PLOT
M-1,N+1
335 LOCATE M,N+1,L:IF L=1 THEN N=N-1:COLOR 0:PLOT M-1,N+
1:J=1
336 IF J=1 THEN 420
340 GOTO 309
420 COLOR Q:PLOT M,N:COLOR 0:PLOT M-1,N:SOUND 0,M,10,4
425 IF M=K THEN 460
430 FOR A=1 TO Z:NEXT A:M=M+1:GOTO 420
455 IF N=22 THEN 500
460 COLOR Q:PLOT M,N:COLOR 0:PLOT M-1,N-1:SOUND 0,M,10,4
462 IF M=19 AND N<22 THEN 700
465 IF N=20 THEN LOCATE M,N+1,L:IF L=1 THEN 800
470 FOR A=1 TO Z:NEXT A:M=M+1:N=N+1:GOTO 455
500 IF M>BUS+4 THEN COLOR 0:PLOT M-1,N-1:GOTO 800
501 IF M=BUS+3 THEN C=C+200:BO=1
502 SOUND 0,200,2,8
505 Q=169:COLOR 0:PLOT M-1,N-1:COLOR Q:PLOT M,N:FOR A=1
TO 40:NEXT A:SOUND 0,180-INT(P)*7,2,2:Q=165
510 FOR M=M TO 19:COLOR Q:PLOT M,N:COLOR 0:PLOT M-1,N:FO
R A=1 TO Z
520 NEXT A:NEXT M:FOR A=1 TO Z/2:NEXT A:COLOR 0:PLOT 19,
22:SOUND 0,0,0,0
530 C=C+((BUS-3)*10)+100
532 POSITION 14,0:? #6;C
535 IF C>HI THEN HI=C
540 IF BO=1 THEN SOUND 1,15,10,6:FOR A=1 TO 50:NEXT A:PO
SITION 14,2:? #6;"BONUS":SOUND 1,0,0,0:BO=0
550 IF C>3000 AND EB=1 THEN 750
600 IF U=0 THEN 900
610 FOR A=1 TO 500:NEXT A:? #6;CHR$(125):GOTO 80
700 COLOR 0:PLOT M,N
710 FOR A=150 TO 250:SOUND 0,A,2,10:NEXT A:SOUND 0,0,0,0
:U=U-1:GOTO 600
750 U=U+1:COLOR 165:PLOT U-1,0:SOUND 0,20,10,6:FOR A=1 T
O 30:NEXT A:SOUND 0,0,0,0:EB=0:GOTO 535
800 COLOR 167:PLOT M,N:FOR A=150 TO 240:SOUND 0,A,2,8
805 IF A=180 THEN COLOR 168:PLOT M,N
```

STUNT RIDER

```
810 NEXT A:SOUND 0,0,0,0:U=U-1:GOTO 600
850 FOR T=1 TO 15:READ E:SOUND 2,E,10,10:SOUND 3,E+1,10,
8:FOR A=1 TO 25:NEXT A:NEXT T:SOUND 2,0,0,0:SOUND 3,0,0,
0
860 FOR A=1 TO 18:NEXT A
870 FOR T=1 TO 30:READ E:SOUND 2,E,10,10:SOUND 3,E+1,10,
8:FOR A=1 TO 10:NEXT A:NEXT T:SOUND 2,0,0,0:SOUND 3,0,0,
0
875 FOR A=0 TO 10:READ B:POKE 1536+A,B:NEXT A:POKE 512,0
:POKE 513,6:T=PEEK(560)+256*PEEK(561):POKE T+27,134
885 RESTORE 1100
890 GOTO 74
900 POSITION 5,4:? #6;"game over":POSITION 4,17:? #6;"HI
-SCORE ";HI
905 POSITION 3,12:? #6;"press trigger":POSITION 3,13:? #
6;"to play again"
910 IF STRIG(0)=0 THEN RESTORE 1090:GOTO 15
915 FOR G=0 TO 6:POKE 708,G
920 NEXT G:GOTO 910
1000 DATA 0,0,0,0,0,0,0,0
1010 DATA 124,254,198,130,130,254,254,254
1011 DATA 254,66,66,66,126,254,68,68
1020 DATA 1,2,4,8,16,32,64,128
1030 DATA 255,255,0,0,0,0,0,0
1040 DATA 12,2,236,28,90,165,165,66
1050 DATA 26,29,53,82,160,80,80,32
1060 DATA 137,66,32,8,4,28,119,28
1070 DATA 0,0,0,8,4,28,119,28
1080 DATA 0,12,2,236,90,189,165,66
1090 DATA 12,2,236,28,90,165,165,66
1100 DATA 130,0,140,0,89,89,89,0,130,0,140,0,89,89,89,98
,0,105,0,98,0,105,0,100,0,110,110,0,0,0,90,90,0,0,80,80
1110 DATA 0,0,74,74,74,74,74,74,0
1150 DATA 72,169,196,141,10,212,141,26,208,104,64
```

DID YOU KNOW . . .

This is a new section in the newsletter which will be devoted to those interesting "hot tips" and useful PEEKS & POKES. So if you know of any such tips, write in to ROY, at the usual address and we will pass them on to other members.

Please note for this section, if submitting a small program, we must limit the line length to approximately 30 lines.

LITTLE BLACK LINES

Did you know that you can generate thin black lines in between the line of text on a graphic zero mode. It does this by re-writing the display list just below the existing one, and inserting a 'BLANK ONE LINE' instruction between each 'MODE TWO LINE' instruction.

For those of you who have not had your ATARI very long, the 'DISPLAY LIST' is actually the program that the video processor chip (ANTIC) runs to decide which modes to place at various parts of the screen, and where to pick up its display data from in memory. Try this program:—

```
5 REM ..Generates Lines On Screen ..
10 OLD=PEEK(561):DL=PEEK(560)+256*OLD
20 DL1=DL-256
30 FOR A=1 TO 6
40 POKE DL1+A,PEEK(DL+A):NEXT A
50 FOR A=6 TO 50 STEP 2
60 POKE DL1+A,0:POKE DL1+1+A,2:NEXT A
70 FOR A=52 TO 53
80 POKE DL1+A,PEEK(DL+A-23):NEXT A
90 POKE DL1+54,PEEK(561)-1
100 POKE 561,OLD-1
```

If you place this in your own program as a subroutine, it only needs to be executed once. To remove the lines you only need to 'POKE 561,OLD' as a program line or in immediate mode. To once again produce the lines you need only 'POKE 561,OLD-1'

DATA STATEMENT GENERATOR

Here is a short utility program to convert a binary load file into DATA statements. The program starts the DATA statement lines from line 32701, and writes eight DATA elements per line number. These values can of course be altered to suit your requirements.

Line 120 exists to dispose of the first six bytes of the binary load file, as these contain the load address and size for use by DOS. This program was originally written to create the data statements in the 'TEXT ON GRAPHICS 8' program, and requires a disk drive.

It is, however, possible to use it with a cassette system if you first load the bytes into a numerical array before printing out the DATA lines.

```
10 REM < DATA STATEMENT GENERATER >
20 REM <<<     By Ron Levy     >>>
100 OPEN #1,4,0,"D:LOADFILE"
110 OPEN #2,8,0,"D:DATAFILE"
120 FOR I=1 TO 6:GET #1,A:NEXT I
130 TRAP 1000:LINE=32701
200 PRINT #2;LINE;" DATA ";
300 FOR I=1 TO 8
350 GET #1,A
360 IF I<>1 THEN PRINT #2;",";
370 PRINT #2;A;
390 NEXT I
400 LINE=LINE+1
410 PRINT #2:GOTO 200
1000 PRINT #2:CLOSE #1:CLOSE #2
```

WORD PROCESSOR??

Did you know you could use your ASSEMBLER/EDITOR Cartridge as a cheap form of word processor, for writing letters etc. Insert the cartridge and go into EDIT mode, enter text lines, with line numbers as per entering source code, remembering that up to 3 screen lines can be entered per line number. Blank lines cannot be entered directly, I use an ESCAPE CONTROL sequence to print blanks such as: 150 ← → RET. Don't worry left and right arrows won't come out on the printer. Printer control characters can be embedded within the text or put on separate line numbers.

The text can be edited using the system edit controls. To save the text with line numbers use LIST#D: FILENAME or without them use PRINT #D: FILENAME. Cassette owners can use the C: device code without a filename. The whole text can be viewed with the PRINT or LIST, the former does not display the line numbers (CONTROL 1 to pause). Text is sent to the printer with the PRINT#P: command, again without line numbers. Text with line numbers can be retrieved with the ENTER#D: FILENAME. Disk users can also use the COPY option 'C' of DOS to send the text to the printer or screen directly, just respond to this option with D: FILENAME, P: (or use E: for screen). No cartridge is required to do this.

When a text file is in memory and under control of the ASSEMBLER/ EDITOR cartridge, all the FIND and REPLACE commands can be used on the text. BASIC programs can be typed in with this cartridge with auto line numbering, the program is then LISTed, as above, to disk or tape, then after changing to the BASIC cartridge, ENTERed (syntax checking is done at this stage) from the disk or tape. Finally the SIZE command returns the address of the text buffer. I imagine that the experienced machine code programmer could add more sophisticated commands with some machine code routines to access and modify this buffer.

H. M. Hoffman – London.

SPACE FORTRESS

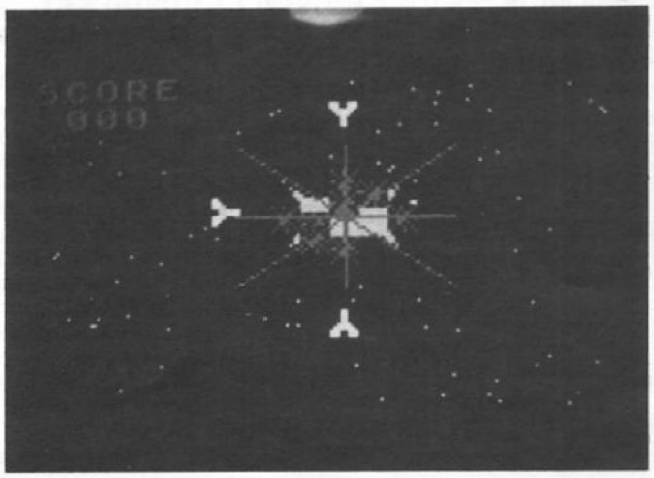
Runs in 32K Cassette or Disk

by Alex Macklen — Truro

Can you save the Space Fortress from the attacking missiles? Your fortress can fire in four directions to shoot down the enemy. This game is for one player and uses a joystick in port one. There is no need to push the fire button, just push the joystick in the direction you wish to fire. When you first 'RUN' the game, the screen will rotate through lots of colours for about a minute, before the game starts.

NOTE: In this program, anything which is underlined, should be entered in "INVERSE"

```
1 GOSUB 32000:CLR
10 GOSUB 20000
20 GRAPHICS 7+16:POKE 756,PEEK(106)+1
30 SETCOLOR 2,3,4:SC=0:J1=1:MX1=0:MY1=0:MX2=0:MY2=0:MX3=
0:MY3=0:MX4=0:MY4=0:M1=0:M2=0:M3=0:M4=0
40 COLOR 1:FOR Y=35 TO 45:PLOT 72,Y:DRAWTO 95,Y:NEXT Y
50 PLOT 72,35:DRAWTO 69,32:PLOT 73,35:DRAWTO 69,32:PLOT
72,36:DRAWTO 69,32
60 PLOT 72,45:DRAWTO 69,48:PLOT 73,45:DRAWTO 69,48:PLOT
72,44:DRAWTO 69,48
70 PLOT 95,35:DRAWTO 98,32:PLOT 94,35:DRAWTO 98,32:PLOT
95,36:DRAWTO 98,32
80 PLOT 95,45:DRAWTO 98,48:PLOT 94,45:DRAWTO 98,48:PLOT
95,44:DRAWTO 98,48
90 FOR STAR=1 TO 80:STARX=RND(0)*158+1:STARY=RND(0)*94+1
:PLOT STARX,STARY:NEXT STAR
100 COLOR 0:FOR X=73 TO 94 STEP 2:PLOT X,40:NEXT X
110 D=USR(1536,83,3,0,0):D=USR(1536,67,3,1,0):D=USR(1536
,79,3,2,0)
120 D=USR(1536,82,3,3,0):D=USR(1536,69,3,4,0):D=USR(1536
,48,3,1,8):D=USR(1536,48,3,2,8):D=USR(1536,48,3,3,8)
130 ON J1 GOTO 140,150,160,170
140 D=USR(1536,32,1,70,24):GOTO 180
150 D=USR(1536,32,1,72,34):GOTO 180
160 D=USR(1536,32,1,70,43):GOTO 180
170 D=USR(1536,32,1,68,34)
180 J=STICK(0):IF J=15 THEN GOTO 290
190 IF J=10 OR J=14 OR J=6 THEN J1=1:D=USR(1536,16,1,70,
24):GOTO 230
200 IF J=7 THEN J1=2:D=USR(1536,17,1,72,34):GOTO 230
210 IF J=5 OR J=13 OR J=9 THEN J1=3:D=USR(1536,18,1,70,4
3):GOTO 230
220 IF J=11 THEN J1=4:D=USR(1536,19,1,68,34)
230 COLOR 1:SOUND 0,25,10,8:SOUND 1,28,10,8:ON J1 GOTO 2
50,260,270,280
250 PLOT 84,27:DRAWTO 84,0:COLOR 0:PLOT 84,27:DRAWTO 84,
0:IF M1=1 THEN M1=0:D=USR(1536,15,3,MX1,MY1):SC=SC+2
255 GOTO 290
260 PLOT 104,40:DRAWTO 159,40:COLOR 0:PLOT 104,40:DRAWTO
 159,40:IF M2=1 THEN M2=0:D=USR(1536,15,3,MX2,MY2):SC=SC
+2
265 GOTO 290
270 PLOT 84,54:DRAWTO 84,95:COLOR 0:PLOT 84,54:DRAWTO 84
,95:IF M3=1 THEN M3=0:D=USR(1536,15,3,MX3,MY3):SC=SC+2
275 GOTO 290
280 PLOT 63,40:DRAWTO 0,40:COLOR 0:PLOT 63,40:DRAWTO 0,4
0:IF M4=1 THEN M4=0:D=USR(1536,15,3,MX4,MY4):SC=SC+2
290 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUND 3,0,0,0:IF SC>999
THEN GOTO 700
```



```
300 V1=INT(SC/100):V2=INT(SC/10-V1*10):V3=SC-V1*100-V2*1
0:V1=V1+48:V2=V2+48:V3=V3+48
310 D=USR(1536,V1,3,1,8):D=USR(1536,V2,3,2,8):D=USR(1536
,V3,3,3,8)
320 IF M1=0 THEN IF INT(RND(0)*2+1)=1 THEN M1=1:MX1=70:M
Y1=SC/75:D=USR(1536,20,2,MX1,MY1)
330 IF M2=0 THEN IF INT(RND(0)*2+1)=1 THEN M2=1:MX2=79-S
C/400:MY2=33:D=USR(1536,21,2,MX2,MY2)
340 IF M3=0 THEN IF INT(RND(0)*2+1)=1 THEN M3=1:MX3=70:M
Y3=70-SC/75:D=USR(1536,22,2,MX3,MY3)
350 IF M4=0 THEN IF INT(RND(0)*2+1)=1 THEN M4=1:MX4=60+S
C/400:MY4=32:D=USR(1536,23,2,MX4,MY4)
360 IF M1=1 THEN D=USR(1536,20,0,MX1,MY1):MY1=MY1+1:D=US
R(1536,20,2,MX1,MY1):IF MY1>=24 THEN GOTO 500
370 IF M2=1 THEN D=USR(1536,21,0,MX2,MY2):MX2=MX2-1:D=US
R(1536,21,2,MX2,MY2):IF MX2<=72 THEN GOTO 500
380 IF M3=1 THEN D=USR(1536,22,0,MX3,MY3):MY3=MY3-1:D=US
R(1536,22,2,MX3,MY3):IF MY3<=43 THEN GOTO 500
390 IF M4=1 THEN D=USR(1536,23,0,MX4,MY4):MX4=MX4+1:D=US
R(1536,23,2,MX4,MY4):IF MX4>=68 THEN GOTO 500
400 GOTO 130
500 SOUND 0,50,8,8:SOUND 1,100,8,8:SOUND 2,200,8,8:SOUND
 3,5,8,8
510 D=USR(1536,15,3,68,34):D=USR(1536,15,3,70,43):D=USR(
1536,15,3,72,34):D=USR(1536,15,3,70,24)
520 D=USR(1536,15,3,69,36):D=USR(1536,15,3,69,40):D=USR(
1536,15,3,70,30):D=USR(1536,15,3,71,27)
530 FOR N=0 TO 3:SOUND N,0,0,0:NEXT N
540 FOR N=1 TO 150:NEXT N
550 FOR N=0 TO 3:SOUND N,N*80+5,8,8:NEXT N
560 COLOR 3:PLOT 84,40:DRAWTO 84,20:DRAWTO 84,60:PLOT 84
,40:DRAWTO 114,40:DRAWTO 54,40:PLOT 84,40:DRAWTO 114,20
570 PLOT 84,40:DRAWTO 114,60:PLOT 84,40:DRAWTO 54,60:PLO
T 84,40:DRAWTO 54,20
580 FOR W=0 TO 15:FOR W1=1 TO 100:SETCOLOR 2,W,5:NEXT W1
:NEXT W
590 FOR N=0 TO 3:SOUND N,0,0,0:NEXT N
600 GRAPHICS 2+16:? #6;"GAME OVER..FINAL":? #6;"SCORE ";
SC:? #6;"TO PLAY AGAIN":? #6;"PRESS TRIGGER"
```

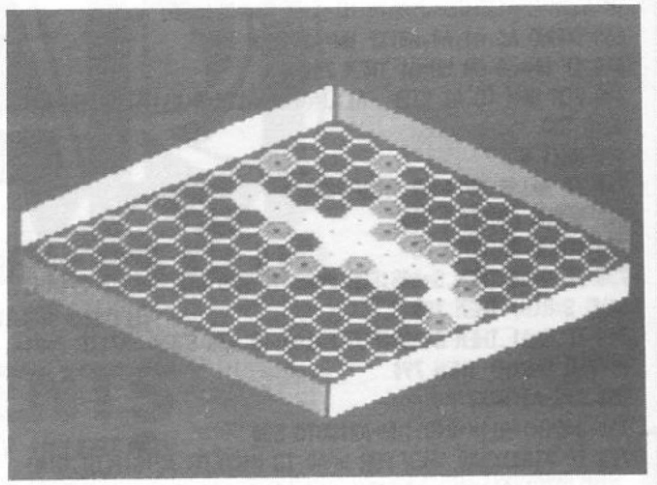
Continued on 28

Runs in 16K Cassette or 32K Disk

HEX

by Ezio Bottarelli and Antonio Tocchi — Italy

This program is not as serious as it's name suggests. It is in fact a game for two playes, using joysticks in ports one and two. An hexagonal grid is drawn in pleasing shades of blue and pink, and the object of the game is to join the diagonally opposite sides of the playing area with a line of hexagons in your chosen colour, before your opponent can do the same. This game can develop into a game of wits, as you decide to either block your opponents line or make a run for home.



```
19 GRAPHICS 0:POSITION 5,7:POKE 82,5:POKE 752,1
20 ? "HHH  HHH  EEEEEEE XXXX    XXXX"
30 ? "HHH  HHH  EEEEEEE  XXXX  XXXX"
40 ? "HHH  HHH  EEE       XXXXXXXX"
50 ? "HHHHHHHH  EEEEEEE    XXXXXX"
60 ? "HHHHHHHH  EEEEEEE    XXXXXX"
70 ? "HHH  HHH  EEE       XXXXXXXX"
80 ? "HHH  HHH  EEEEEEE  XXXX  XXXX"
90 ? "HHH  HHH  EEEEEEE XXXX    XXXX"
100 ? :? :? " by A. Tocchi & E. Bottarelli"
110 FOR T=40 TO 256:SETCOLOR 2,T/17,4:SETCOLOR 1,T,T:SOU
ND 0,T,10,6
115 SOUND 1,T-2,10,6:SOUND 2,T-4,10,6:FOR D=1 TO 2:NEXT
D:NEXT T
116 GOSUB 1000
120 DIM Q(60),R(60)
130 A0=0:A1=1:A2=2:A3=3:A4=4:A6=6:A7=7:A8=8:A9=9:A10=10:
A12=12:A18=18:A48=48:A50=50:A78=78:AA=730:AB=770:AC=210
135 SOUND A0,A0,A0,A0:SOUND A1,A0,A0,A0:SOUND A2,A0,A0,A
0
140 GRAPHICS 23:COLOR A1:X=74:Y=74:Z=A7:SETCOLOR 1,3,10
150 FOR B=A18 TO A78 STEP A3:FOR A=X TO Y STEP 14:PLOT A
,B:DRAWTO A+A3,B-A3:DRAWTO A+A7,B-A3:DRAWTO A+A10,B
160 DRAWTO A+A7,B+A3:DRAWTO A+A3,B+A3:DRAWTO A,B:NEXT A:
X=X-Z:Y=Y+Z:IF X=A4 THEN Z=-A7
170 NEXT B
180 A=46:B=A12:FOR R=A1 TO A9:COLOR A2:PLOT A0,A:DRAWTO
A78,B:COLOR A3:PLOT 158,A:DRAWTO 80,B:A=A-A1:B=B-A1:NEXT
 R
190 A=50:B=84:FOR R=A1 TO A9:COLOR A3:PLOT A0,A:DRAWTO A
78,B:COLOR A2:PLOT 158,A:DRAWTO 80,B:A=A+A1:B=B+A1:NEXT
R
200 Q(A0)=A0:R(A0)=A0:X=79:Y=51:SOUND A0,A12,A12,A10:GOT
O 390
210 SOUND A0,A12,A12,A10:IF INT(C/A2)*A2=C THEN G=A0:GOT
O 230
220 G=A1
230 Z=STICK(G):POKE 77,A0
240 COLOR A0:PLOT X,Y:FOR T=A0 TO A50:NEXT T
250 X=X+((Z=5)+(Z=A6)+(Z=A7))*A7
260 X=X-((Z=A9)+(Z=A10)+(Z=11))*A7
270 IF X<A9 THEN X=A9
280 IF X>149 THEN X=149
290 Y=Y+((Z=5)+(Z=A9)+(Z=13))*A6
300 Y=Y-((Z=A6)+(Z=A10)+(Z=14))*A6
310 IF Z=15 THEN 330
320 SOUND A0,A50,A10,A10
330 IF Y<A18 THEN Y=A18
340 IF Y>A78 THEN Y=A78
350 IF X=AX THEN 380
360 IF INT(X/A2)*A2=X THEN Y=Y+A3
370 AX=X
380 IF X=BX THEN 410
390 IF INT(X/A2)*A2<>X THEN Y=Y-A3
400 BX=X
410 LOCATE X,Y,L:IF L>A0 THEN X=VX:Y=VY
420 COLOR A1:PLOT X,Y:FOR T=A0 TO A50:NEXT T:SOUND A0,A0
,A0,A0:VX=X:VY=Y
430 LOCATE X-A3,Y,L:IF L>A0 THEN 230
440 IF STRIG(G)=A0 THEN COLOR A2+G:PLOT X-A2,Y-A2:DRAWTO
 X+A2,Y-A2:PLOT X-A3,Y-A1:DRAWTO X+A3,Y-A1:GOTO 460
450 GOTO 230
460 PLOT X-A4,Y:DRAWTO X+A4,Y:PLOT X-A3,Y+A1:DRAWTO X+A3
,Y+A1:PLOT X-A2,Y+A2:DRAWTO X+A2,Y+A2
470 SOUND A0,20,A10,A10:C=C+A1:HH=A0+G
480 IF G=A0 THEN B=A78:FOR A=79 TO 149 STEP A7:LOCATE A-
A1,B,L:IF L=A2 THEN COLOR 0:PLOT A-A1,B:Q(H)=A-A1:R(H)=B
:GOTO 520
490 IF G=A1 THEN B=A78:FOR A=79 TO A9 STEP -A7:LOCATE A-
A1,B,L:IF L=A3 THEN COLOR 0:PLOT A-A1,B:Q(H)=A-A1:R(H)=B
:GOTO 520
500 B=B-A3:HH=A0+G:NEXT A
510 GOSUB AB:GOTO AC
520 J=A:K=B
530 IF G=A0 THEN V=A48:FOR O=149 TO 79 STEP -A7:IF J=O A
ND K=V THEN HH=A0
540 IF G=A1 THEN V=A48:FOR O=A9 TO 79 STEP A7:IF J=O AND
 K=V THEN HH=A1
550 V=V+A3:NEXT O
560 IF HH=A0 THEN 600
570 IF HH=A1 THEN 610
580 LOCATE J-A1,K+A6,L:IF L=A2+G THEN COLOR A0:PLOT J-A1
,K+A6:H=H+A1:Q(H)=J-A1:R(H)=K+A6:WW=A0:GOTO AA
590 LOCATE J-A8,K+A3,L:IF L=A2+G THEN COLOR A0:PLOT J-A8
,K+A3:H=H+A1:Q(H)=J-A8:R(H)=K+A3:WW=A0:GOTO AA
600 LOCATE J-A8,K-A3,L:IF L=A2+G THEN COLOR A0:PLOT J-A8
,K-A3:H=H+A1:Q(H)=J-A8:R(H)=K-A3:WW=A0:GOTO AA
610 LOCATE J-A1,K-A6,L:IF L=A2+G THEN COLOR A0:PLOT J-A1
,K-A6:H=H+A1:Q(H)=J-A1:R(H)=K-A6:WW=A0:GOTO AA
620 IF HH=A0 THEN 660
630 LOCATE J+A6,K-A3,L:IF L=A2+G THEN COLOR A0:PLOT J+A6
,K-A3:H=H+A1:Q(H)=J+A6:R(H)=K-A3:WW=A0:GOTO AA
```

HEX

```
640 IF HH=A1 THEN 660
650 LOCATE J+A6,K+A3,L:IF L=A2+G THEN COLOR A0:PLOT J+A6
,K+A3:H=H+A1:Q(H)=J+A6:R(H)=K+A3:WW=A0:GOTO AA
660 SOUND A0,A0,A0,A0:IF WW=A1 THEN 690
670 IF HH=A0 OR HH=A1 THEN 700
680 FOR W=H TO A1 STEP -A1:J=Q(W)+A1:K=R(W):WW=A1:HH=A2:
GOTO 530
690 NEXT W
700 IF A=149 OR A=A9 THEN 720
710 GOSUB AB:GOTO 500
720 GOSUB AB:GOTO AC
730 IF G=A0 THEN S=A48:FOR P=A8 TO A78 STEP A7:IF P=Q(H)
 AND S=R(H) THEN 790
740 IF G=A1 THEN S=A48:FOR P=148 TO A78 STEP -A7:IF P=Q(
H) AND S=R(H) THEN 790
750 S=S-A3:NEXT P
760 J=Q(H)+A1:K=R(H):HH=A2:GOTO 530
770 IF Q(A0)<>A0 THEN FOR W=A0 TO H:COLOR A2+G:PLOT Q(W)
,R(W):Q(W)=A0:R(W)=A0:NEXT W:H=A0:HH=A0:WW=A0
780 RETURN
790 FOR T=A0 TO 100:SOUND A0,T,A8,15:NEXT T:FOR T=255 TO
 A0 STEP -A1:SOUND A0,T,A10,15:FOR R=A0 TO A2:NEXT R:NEX
T T
795 GRAPHICS 39:POKE 752,1:POKE 710,0:? :? "      P R E
S S   S T A R T"
800 SETCOLOR A1+G,A3+6*G,A4:FOR T=A0 TO A50:NEXT T:SETCO
LOR A1+G,A3+6*G,A10:FOR T=A0 TO A50:NEXT T
810 IF PEEK(53279)<>A6 THEN 800
820 CLR :GOTO 120
1000 GRAPHICS 0:POKE 710,0:POKE 82,2:? :? :?
1010 ? "Game for two players. "
1020 ? "Use 2 joysticks."
1030 ? :? :? "Each player must link the opposite":? "sid
es of the rhomb with his own ":? "pieces."
1060 POKE 752,1:POSITION 12,22:? "press any key"
1070 POKE 764,255
1080 IF PEEK(764)=255 THEN 1080
1090 RETURN
```

Continued from 26

SPACE FORTRESS

```
610 IF STRIG(0)=1 THEN GOTO 610
620 GOTO 20
700 GRAPHICS 2+16:? #6;"GOOD GAME!!!":? #6;"YOU WON!!!":
? #6;"YOUR SPACE FORTRESS":? #6;"SURVIVED"
710 ? #6;"TO PLAY AGAIN":? #6;"PRESS TRIGGER":GOTO 610
19999 END
20000 ML=1536:FOR I=0 TO 252:READ A:POKE ML+I,A:NEXT I:R
ETURN
20010 DATA 104,240,10,201,4,240
20020 DATA 11,170,104,104,202,208
20030 DATA 251,169,253,76,164,246
20040 DATA 104,133,195,104,201,128
20050 DATA 144,4,41,127,198,195
20060 DATA 170,141,250,6,224,96
20070 DATA 176,15,169,64,224,32
20080 DATA 144,2,169,224,24,109
20090 DATA 250,6,141,250,6,104
20100 DATA 104,141,251,6,104,104
20110 DATA 141,252,6,14,252,6
20120 DATA 104,104,141,253,6,133
20130 DATA 186,166,87,169,10,224
20140 DATA 3,240,8,169,20,224
20150 DATA 5,240,2,169,40,133
20160 DATA 207,133,187,165,88,133
20170 DATA 203,165,89,133,204,32
20180 DATA 228,6,24,173,252,6
20190 DATA 101,203,133,203,144,2
20200 DATA 230,204,24,165,203,101
20210 DATA 212,133,203,165,204,101
20220 DATA 213,133,204,173,250,6
20230 DATA 133,187,169,8,133,186
20240 DATA 32,228,6,165,212,133
20250 DATA 205,173,244,2,101,213
20260 DATA 133,206,160,0,162,8
20270 DATA 169,0,133,208,133,209
20280 DATA 177,205,69,195,72,104
20290 DATA 10,72,144,8,24,173
20300 DATA 251,6,5,208,133,208
20310 DATA 224,1,240,8,6,208
20320 DATA 38,209,6,208,38,209
20330 DATA 202,208,228,104,152,72
20340 DATA 160,0,165,209,145,203
20350 DATA 200,165,208,145,203,104
20360 DATA 168,24,165,203,101,207
20370 DATA 133,203,144,2,230,204
20380 DATA 200,192,8,208,183,96
20390 DATA 169,0,133,212,162,8
20400 DATA 70,186,144,3,24,101
20410 DATA 187,106,102,212,202,208
20420 DATA 243,133,213,96,0,1
20430 DATA 28
32000 POKE 106,PEEK(106)-16:GRAPHICS 0:START=(PEEK(106)+
1)*256:POKE 756,START/256:POKE 752,1
32020 FOR Z=0 TO 1023:POKE START+Z,PEEK(57344+Z):SETCOLO
R 2,0,RND(0)*255+1:NEXT Z:RESTORE 32100
32030 READ X:IF X=-1 THEN RESTORE :RETURN
32040 FOR Y=0 TO 7:READ Z:POKE X+Y+START,Z:NEXT Y:GOTO 3
2030
32100 DATA 632,145,82,44,222,57,52,74,137
32101 DATA 640,24,24,24,60,126,126,60,255
32102 DATA 648,128,176,248,255,255,248,176,128
32103 DATA 656,255,60,126,126,60,24,24,24
32104 DATA 664,1,13,31,255,255,31,13,1
32105 DATA 672,231,231,126,60,24,24,24,0
32106 DATA 680,3,7,15,252,252,15,7,3
32107 DATA 688,24,24,24,24,60,126,231,231
32108 DATA 696,192,224,240,63,63,240,224,192
32109 DATA -1
```

THE U.K.

ATARI

COMPUTER OWNERS CLUB

INDEPENDENT USER GROUP

The U.K. ATARI COMPUTER OWNERS CLUB, P.O. BOX 3, Rayleigh, Essex.