PRICE £1.00

THE U.K.

ATARI

COMPUTER OWNERS CLUB

INDEPENDENT USER GROUP

ISSUE 6

THE U.K. ATARI COMPUTER OWNERS CLUB

INDEPENDENT USER GROUP

CONTENTS

THE RIGHT ONE?

In recent months you may have noticed a fall in the number of adverts in the computer magazines for Atari software and hardware. This is mainly due to the fact that the Commodore, Spectrum and B.B.C. computers are selling better in this country than the Atari. The popularity of a computer is not necessarily a measure of it's performance. A well planned advertising campaign can often convince people how fantastic a computer is, when in reality it's performance falls well short of the impression created. This is *not* the case for the Atari, since the adverts have been poor and so rare, as to make it an unpopular machine, as far as the masses are concerned. The Atari sells to people who can recognize a high quality computer with a superior performance.

The membership of the user group is growing, and on average there have been one hundred new members a month, since the last Newsletter.

In this issue is the second part of the series 'Cracking the Code', a deeper look into machine code programming for beginners; a simple yet very effective light pen circuit for your Atari, including some programs to test it. We hope you have fun building the circuit and that we shall see light pen programs from you in future issues. A new feature to the Newsletter is the Adventure Column, offering tips on solving, and writing Adventure Games. Also, we review some of the latest programs to be released; 'Java Jim', a new British produced game by Thorn E.M.I., and 'Fun With Art', a powerful graphics composer in cartridge format. If you own an Epson or Centronics printer with a graphic mode, then 'Get the Picture', a program and explanation of how to print out a hi-res graphics eight screen is for you. But for those of you without a printer, but still on graphics eight, we offer a clever program that gives you an eighty column text display. The program in the article 'On Line to the World' provides a simple radio tele-type receive and transmit function when used with the Atari 850 interface module, and a radio signal terminal unit.

Finally, we would like to thank all those members who have contributed to the software library since the last issue, all the programs have been of a very high standard.

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.

GET THE PICTURE!

by Keith Mayhew & Roy Smith

Do you own a dot-matrix printer which is capable of printing graphic images? Are you the artistic type who loves to create intricate pictures? Or maybe you just wish to print graphs or charts or even circuit diagrams. This article will provide you with the means to transfer your creations to a Centronics 739 or an Epson capable of graphics (MX80 F/T III, RX80, FX80, etc.) with high speed and relative ease. If you own a different printer, by reading this article and slightly modifying the programs given, you should be able to adapt it to your own needs.

First, we will give a brief description of how graphic images are stored in the computer's memory. A graphics 8 screen is made up of 320 by 192 pixels and in this mode the foreground 'colour' is only a shade of the background colour, as in the standard text mode. This means that only one bit is needed per pixel, on or off. Thus, in one byte eight pixels are accommodated, so 40*192 bytes are needed for the whole display, i.e. approximately 8K of display RAM. The other commonly used mode for graphics is a mode not used by the 400/800 operating system (available direct to the user on the new 'XL' range) which is ANTIC mode 14. In this mode it uses two bits per pixel so that one of three colour registers can be selected, giving a four colour display with the background. The use of two bits per pixel means to accommodate the picture in the same amount of RAM the horizontal resolution is halved to 160 pixels. Both of these modes have utilities that you can buy or type in to draw on them. For the graphics 8 pictures shown in this article, 'Versawriter' was used which is a hardware/software tool configured as a drawing board with a moving arm and a magnifier. For the multi-colour mode 'Micropainter' was used, which is a joystick controlled software package with very powerful features. Of course, these are only utilities and any picture which can be accessed in these modes can also be printed, i.e. produced by BASIC's PLOT/DRAWTO commands or maybe 'GRAPH-IT' an ATARI program for producing pie charts, bar charts, polar plots, 2D and 3D plots. These can either be modified to save the finished screen for further use or the printer program can be 'tacked on' as an extra facility.

The Centronics 739 printer, when switched by software into graphics mode, accepts data in a six bit form and an offset of 32 is added to each byte so that the data ranges from 32 to 95. The only problem is that the computer stores the data in a horizontal structure and the printer accepts in a vertical pattern. So in order to reproduce the screen picture the first six lines are used as a block. To calculate the byte the printer wants the first bit of each byte is put together to form a six bit word (plus the 32 offset), this process is then continued for eight bits to complete one whole byte and again repeated for all 40 bytes in the line. This program was firstly written in BASIC but took literally half an hour to run! By replacing the time consuming part, described above, with machine code and calling it as a subroutine from BASIC the screen is printed almost as fast as the printer can handle the data. The Epson works on the same principle except the data is of an eight bit format not six, so the sampled block would have to be eight lines deep.

It must be remembered that the Epson high-res mode uses codes from Ø to 255, which means you must have an eight bit interface, also one problem which is encountered because of this is that the code of 155, representing a certain bit pattern, will be converted by the ATARI into a code of 13, which is the ASCII carriage return code. As you can imagine this could cause a problem by printing the wrong pattern within a picture, but on the other hand the likelihood of 155 being transmitted is extremely small. This is why the Centronics uses only six bits to convey the data. Other considerations when using the Epson printers are that the line spacing has to be adjusted and the graphics command must be sent for every line printed, whereas on the Centronics there is no adjustment required for line spacing and the graphics mode is only sent once at the start.

Listing 1 is the assembly language program for dumping a picture via BASIC for the Centronics 739 printer, listing 2 is a

```
0100 ;High resolution printer subroutine.
0110 ;Called by BASIC for CENTRONICS 739 printers.
0120 ;Written by Keith Mayhew.
0130         *=      $0600
0140 SAVMSC  =       $58        Start of screen.
0150 PZ0     =       $CB        Temporary pointer.
0160         PLA                Clean stack.
0170         PLA                Screen offset high.
0180         TAY                Save in 'Y'.
0190         PLA                Screen offset low.
0200         CLC                Add to
0210         ADC     SAVMSC     screen start
0220         STA     PZ0        and store it.
0230         TYA                Get high byte.
0240         ADC     SAVMSC+1   Add to screen
0250         STA     PZ0+1      and store it.
0260         PLA                String high.
0270         STA     STRING+2   Save it.
0280         PLA                String low.
0290         STA     STRING+1   Save it.
0300         LDY     #$00       Index to string.
0310 LOOP4   LDX     #$00       Index to registers.
0320 LOOP3   LDA     (PZ0),Y    Load byte
0330         STA     REG,X      and save it.
0340         LDA     PZ0        Add 40
0350         CLC                to point
0360         ADC     #40        to next
0370         STA     PZ0        line.
0380         BCC     SKIP2      Add to
0390         INC     PZ0+1      high byte.
0400 SKIP2   INX                Next register.
0410         CPX     #$06       Last one.
0420         BNE     LOOP3      No - go back.
0430         LDA     PZ0        Point
0440         SEC                back
0450         SBC     #240&$FF   to
0460         STA     PZ0        the
0470         LDA     PZ0+1      start
0480         SBC     #240/256   of
0490         STA     PZ0+1      block.
0500         LDA     #$00       Set count
0510         STA     BIT        to zero.
0520 LOOP2   LDA     #$00       Clear 'A'.
```

```
0530          LDX     #$05      Last register.
0540 LOOP1    ASL     REG,X     Shift it left.
0550          ROL     A         Put carry in 'A'.
0560          DEX               Same for
0570          BPL     LOOP1     all registers.
0580          CLC               Add $20
0590          ADC     #$20      for printer.
0600 STRING   STA     $FFFF     Store calaculated byte.
0610          INC     STRING+1  Increment
0620          BNE     SKIP1     string
0630          INC     STRING+2  pointer.
0640 SKIP1    INC     BIT       Do the
0650          LDA     BIT       same
0660          CMP     #$08      for all
0670          BNE     LOOP2     eight bits.
0680          INY               Point to next byte.
0690          CPY     #40       Is it 40?
0700          BNE     LOOP4     No - go back.
0710          RTS               Return to BASIC.
0720 BIT      *=      *+1       Bit counter.
0730 REG      *=      *+6       Temporary shift registers.
```

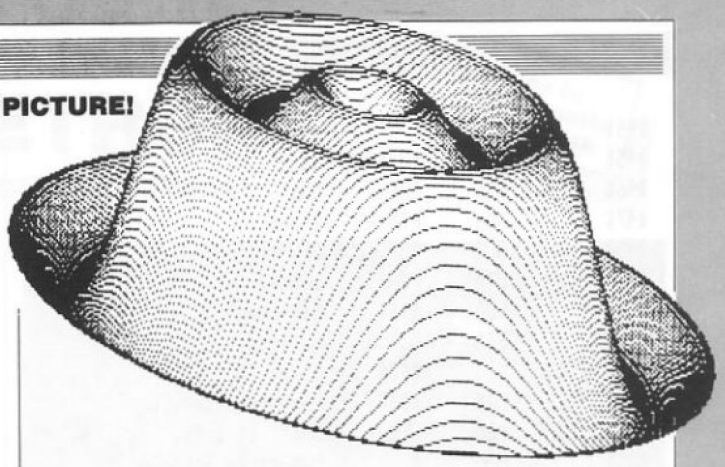
Listing 1. Centronics Machine Code.

similar program for the Epson printers. The way the programs work has been explained above, but for those more familiar with machine code a step by step guide is printed in the form of comments on the listings. To interface between BASIC and the machine code, BASIC must pass certain parameters as it enters the machine code, these are the address of the BASIC string and the number of the block to be encoded. These values change depending on which printer is being used. The string length changes between 240 (Centronics) and 320 (Epson) bytes, because of the different block sizes. It is important that you set the last byte in the BASIC string to a space, so that BASIC has actually allocated a whole block of memory for the string, otherwise the machine code will write the data into the string and BASIC will still think it has no characters within it because the length will still be set to zero. Also, due to the difference in block size, the number for each block will range from Ø to 31 (Centronics) or Ø to 23 (Epson). Note that the string address passed to the Epson program is actually 4 bytes into the string to allow for the graphics command at the start of the string (4 bytes).

The machine code loader program is very simple indeed due to the architecture of the operating system. It uses channel number one for all I/O operations and, to save on the length of code, the channel is opened through BASIC. Then by calling the loader whilst in the desired graphics mode it will find the start of the screen and place that as the buffer address, and places the length of the screen into the buffer length. Then by simply calling the Central I/O system (CIO) and telling it the command is to get bytes, the operating system will return when all the data is read and stored away directly on the screen. If the data was stored on cassette instead of disk then as long as the channel is opened for cassette the loader will operate just the same, giving very fast results. See listing 3 for the program and associated comments.

Listings 4, 5, 6, and 7 are complete BASIC programs for loading and printing versawriter and micropainter pictures. They all incorporate the loader program in the form of 'DATA' statements. Just type in and save from the two programs which are relevant to the printer you use.

All four programs are designed to work on disk systems as the versawriter and micropainter programs are supplied on disk. Any one of the four programs can be modified to work on cassette files, none of the machine code needs to be modified. Remember that the printing part of a program along with its 'DATA' statements can be extracted and put on the end of any hi-res program (cassette or disk) as an extra utility.



```
0100 ;High resolution printer subroutine.
0110 ;Called by BASIC for EPSON printers.
0120 ;written by Keith Mayhew.
0130          *=      $0600
0140 SAVMSC   =       $58       Start of screen.
0150 PZ0      =       $CB       Temporary pointer.
0160          PLA               Clean stack.
0170          PLA               Screen offset high.
0180          TAY               Save in 'Y'.
0190          PLA               Screen offset low.
0200          CLC               Add to
0210          ADC     SAVMSC    screen start
0220          STA     PZ0       and store.
0230          TYA               Get high byte.
0240          ADC     SAVMSC+1  Add to screen
0250          STA     PZ0+1     and store it.
0260          PLA               String high.
0270          STA     STRING+2  Save it.
0280          PLA               String low.
0290          STA     STRING+1  Save it.
0300          LDY     #$00      Index to string.
0310 LOOP4    LDX     #$00      Index to registers.
0320 LOOP3    LDA     (PZ0),Y   Load byte
0330          STA     REG,X     and save it.
0340          LDA     PZ0       Add 40
0350          CLC               to point
0360          ADC     #40       to next
0370          STA     PZ0       line.
0380          BCC     SKIP2     Add to
0390          INC     PZ0+1     high byte.
0400 SKIP2    INX               Next register.
0410          CPX     #$08      Last one?
0420          BNE     LOOP3     No - go back.
0430          LDA     PZ0       Point
```

```
0440          SEC               back
0450          SBC     $320&$FF  to
0460          STA     PZ0       the
0470          LDA     PZ0+1     start
0480          SBC     $320/256  of
0490          STA     PZ0+1     block.
0500          LDA     #$00      Set count
0510          STA     BIT       to zero.
0520 LOOP2    LDA     #$00      Clear 'A'.
0530          LDX     #$07      Last register.
0540 LOOP1    ASL     REG,X     Shift it left.
0550          ROR     A         Put carry into 'A'.
0560          DEX               Same for
0570          BPL     LOOP1     all registers.
0580 STRING   STA     $FFFF     Store calculated byte.
0590          INC     STRING+1  Increment
0600          BNE     SKIP1     string
0610          INC     STRING+2  pointer.
0620 SKIP1    INC     BIT       Do the
0630          LDA     BIT       same
0640          CMP     #$08      for all
0650          BNE     LOOP2     eight bits.
0660          INY               Point to next byte.
0670          CPY     #40       Is it 40?
0680          BNE     LOOP4     No - go back.
0690          RTS               Return to BASIC.
0700 BIT      *=      *+1       Bit counter.
0710 REG      *=      *+8       Temporary shift registers.
```

Listing 2. Epson Machine Code.

```
0100 ;Machine code loader
0110 ;for graphics 8 size screen.
0120 ;Uses channel #1 for I/O.
0130 ;Written by Keith Mayhew.
0140 GETCHR  =       $07
0150 SAVMSC  =       $58
0160 ICCOM   =       $0342
0170 ICBAL   =       $0344
0180 ICBAH   =       $0345
0190 ICBLL   =       $0348
0200 ICBLH   =       $0349
0210 CIOV    =       $E456
0220 LEN     =       192*40
0230         *=      $0680
0240         PLA               Clean stack.
0250         LDX     #$10      Channel #1.
0260         LDA     #GETCHR   Get character.
0270         STA     ICCOM,X   Store command.
0280         LDA     SAVMSC    Screen
0290         STA     ICBAL,X   is
0300         LDA     SAVMSC+1  the
0310         STA     ICBAH,X   buffer.
0320         LDA     #LEN&$FF  Length
0330         STA     ICBLL,X   is 192*40
0340         LDA     #LEN/256  for the
0350         STA     ICBLH,X   screen.
0360         JSR     CIOV      Execute command.
0370         STY     212       Store error
0380         LDY     #$00      status for BASIC.
0390         STY     213       1 is good status.
0400         RTS               Return to BASIC.
```

Listing 3. Machine Code Loader.

```
10 DIM FILE$(14),TEMP$(8),P$(325):GOSUB 1000
20 CONSOL=53279:LPRINT CHR$(27);"%0":P$(325,325)="":CLOS
E #1
100 ? CHR$(125):? "Versawriter picture loader/printer."
110 ? "Enter filename of picture."
120 ? "Do not include 'D:' or '.EXT'"
130 ? "Press start for another picture."
140 ? "Press option for a print out."
150 INPUT TEMP$:L=LEN(TEMP$):FILE$(1,2)="D:":FILE$(3,3+L
)=TEMP$:FILE$(L+3,L+5)=",*"
160 OPEN #1,4,0,FILE$:FOR I=0 TO 13:GET #1,X:NEXT I:GRAP
HICS 24:X=USR(1664)
170 IF PEEK(CONSOL)=6 THEN 20
180 IF PEEK(CONSOL)<>3 THEN 170
190 REM Print out screen image.
200 FOR I=0 TO 31
210 X=USR(1536,I*240,ADR(P$))
220 LPRINT P$:NEXT I
230 GOTO 170
1000 FOR I=0 TO 37:READ D:POKE 1664+I,D:NEXT I
1010 FOR I=0 TO 105:READ D:POKE 1536+I,D:NEXT I
1020 RETURN
1499 REM DATA for loading.
1500 DATA 104,162,16,169,7,157,66,3,165,88,157,68,3,165,
89,157
1510 DATA 69,3,169,0,157,72,3,169,30,157,73,3,32,86,228,
132
1520 DATA 212,160,0,132,213,96
1999 REM CENTRONICS 739 DATA statements.
2000 DATA 104,104,168,104,24,101,88,133,203,152,101,89,1
33,204,104,141
2010 DATA 81,6,104,141,80,6,160,0,162,0,177,203,157,107,
6,165
2020 DATA 203,24,105,40,133,203,144,2,230,204,232,224,6,
208,235,165
2030 DATA 203,56,233,240,133,203,165,204,233,0,133,204,1
69,0,141,106
2040 DATA 6,169,0,162,5,30,107,6,42,202,16,249,24,105,32
,141
2050 DATA 255,255,238,80,6,208,3,238,81,6,238,106,6,173,
106,6
2060 DATA 201,8,208,221,200,192,40,208,175,96
```

Listing 4. BASIC Centronics & Versawriter.

```
10 DIM FILE$(14),TEMP$(8),P$(325):GOSUB 1000
20 CONSOL=53279:LPRINT CHR$(27);"3";CHR$(24):P$(325,325)
="":P$(1,1)=CHR$(27):P$(2,3)="K@":P$(4,4)=CHR$(1)
30 CLOSE #1
100 ? CHR$(125):? "Versawriter picture loader/printer."
110 ? "Enter filename of picture."
120 ? "Do not include 'D:' or '.EXT'"
130 ? "Press start for another picture."
140 ? "Press option for a print out."
150 INPUT TEMP$:L=LEN(TEMP$):FILE$(1,2)="D:":FILE$(3,3+L
)=TEMP$:FILE$(L+3,L+5)=",*"
160 OPEN #1,4,0,FILE$:FOR I=0 TO 13:GET #1,X:NEXT I:GRAP
HICS 24:X=USR(1664)
170 IF PEEK(CONSOL)=6 THEN 20
180 IF PEEK(CONSOL)<>3 THEN 170
190 REM Print out screen image.
200 FOR I=0 TO 23
210 X=USR(1536,I*320,ADR(P$)+4)
```

```
220 LPRINT P$;:NEXT I
230 GOTO 170
1000 FOR I=0 TO 37:READ D:POKE 1664+I,D:NEXT I
1010 FOR I=0 TO 105:READ D:POKE 1536+I,D:NEXT I
1020 RETURN
1499 REM DATA for loading.
1500 DATA 104,162,16,169,7,157,66,3,165,88,157,68,3,165,
89,157
1510 DATA 69,3,169,0,157,72,3,169,30,157,73,3,32,86,228,132
1520 DATA 212,160,0,132,213,96
1999 REM EPSON DATA statements.
2000 DATA 104,104,168,104,24,101,88,133,203,152,101,89,1
33,204,104,141
2010 DATA 78,6,104,141,77,6,160,0,162,0,177,203,157,104,
6,165
2020 DATA 203,24,105,40,133,203,144,2,230,204,232,224,8,
208,235,165
2030 DATA 203,56,233,64,133,203,165,204,233,1,133,204,16
9,0,141,103
2040 DATA 6,169,0,162,7,30,104,6,106,202,16,249,141,255,
255,238
2050 DATA 77,6,208,3,238,78,6,238,103,6,173,103,6,201,8,208
2060 DATA 224,200,192,40,208,178,96,0,0,0
```

Listing 5. BASIC Epson & Versawriter.

```
10 DIM FILE$(14),TEMP$(8),P$(325):GOSUB 1000
20 CONSOL=53279:LPRINT CHR$(27);"%0";P$(325,325)="":CLOS
E #1
100 ? CHR$(125):? "Micropainter picture loader/printer."
110 ? "Enter filename of picture."
120 ? "Do not include 'D:' or '.EXT'"
130 ? "Press start for another picture."
140 ? "Press option for a print out."
150 INPUT TEMP$:L=LEN(TEMP$):FILE$(1,2)="D:":FILE$(3,3+L
)=TEMP$:FILE$(L+3,L+5)=".X"
160 OPEN #1,4,0,FILE$:FOR I=0 TO 39:GET #1,X:NEXT I:GRAP
HICS 24:GOSUB 3000:X=USR(1664)
170 IF PEEK(CONSOL)=6 THEN 20
180 IF PEEK(CONSOL)<>3 THEN 170
190 REM Print out screen image.
200 FOR I=0 TO 31
210 X=USR(1536,I*240,ADR(P$))
220 LPRINT P$;:NEXT I
230 GOTO 170
1000 FOR I=0 TO 37:READ D:POKE 1664+I,D:NEXT I
1010 FOR I=0 TO 105:READ D:POKE 1536+I,D:NEXT I
1020 RETURN
1499 REM DATA for loading.
1500 DATA 104,162,16,169,7,157,66,3,165,88,157,68,3,165,
89,157
1510 DATA 69,3,169,0,157,72,3,169,30,157,73,3,32,86,228,
132
1520 DATA 212,160,0,132,213,96
1999 REM CENTRONICS 739 DATA statements.
2000 DATA 104,104,168,104,24,101,88,133,203,152,101,89,1
33,204,104,141
2010 DATA 81,6,104,141,80,6,160,0,162,0,177,203,157,107,
6,165
2020 DATA 203,24,105,40,133,203,144,2,230,204,232,224,6,
208,235,165
2030 DATA 203,56,233,240,133,203,165,204,233,0,133,204,1
69,0,141,106
2040 DATA 6,169,0,162,5,30,107,6,42,202,16,249,24,105,32
,141
2050 DATA 255,255,238,80,6,208,3,238,81,6,238,106,6,173,
106,6
2060 DATA 201,8,208,221,200,192,40,208,175,96
2999 REM Build a new display list.
3000 DL=PEEK(560)+256*PEEK(561)
3010 POKE DL+3,78:POKE DL+99,78
3020 FOR I=6 TO 98:POKE DL+I,14:NEXT I
3030 FOR I=102 TO 198:POKE DL+I,14:NEXT I
3040 RETURN
```

Listing 6. BASIC Centronics & Micropainter.

```
10 DIM FILE$(14),TEMP$(8),P$(325):GOSUB 1000
20 CONSOL=53279:LPRINT CHR$(27);"3";CHR$(24);P$(325,325)
="";P$(1,1)=CHR$(27):P$(2,3)="K@":P$(4,4)=CHR$(1)
30 CLOSE #1
100 ? CHR$(125):? "Micropainter picture loader/printer."
110 ? "Enter filename of picture."
120 ? "Do not include 'D:' or '.EXT'"
130 ? "Press start for another picture."
140 ? "Press option for a print out."
150 INPUT TEMP$:L=LEN(TEMP$):FILE$(1,2)="D:":FILE$(3,3+L
)=TEMP$:FILE$(L+3,L+5)=".X"
160 OPEN #1,4,0,FILE$:FOR I=0 TO 39:GET #1,X:NEXT I:GRAP
HICS 24:GOSUB 3000:X=USR(1664)
170 IF PEEK(CONSOL)=6 THEN 20
180 IF PEEK(CONSOL)<>3 THEN 170
190 REM Print out screen image.
200 FOR I=0 TO 23
210 X=USR(1536,I*320,ADR(P$)+4)
220 LPRINT P$;:NEXT I
230 GOTO 170
1000 FOR I=0 TO 37:READ D:POKE 1664+I,D:NEXT I
1010 FOR I=0 TO 105:READ D:POKE 1536+I,D:NEXT I
1020 RETURN
1499 REM DATA for loading.
1500 DATA 104,162,16,169,7,157,66,3,165,88,157,68,3,165,
89,157
1510 DATA 69,3,169,0,157,72,3,169,30,157,73,3,32,86,228,
132
1520 DATA 212,160,0,132,213,96
1999 REM EPSON DATA statements.
2000 DATA 104,104,168,104,24,101,88,133,203,152,101,89,1
33,204,104,141
2010 DATA 78,6,104,141,77,6,160,0,162,0,177,203,157,104,
6,165
2020 DATA 203,24,105,40,133,203,144,2,230,204,232,224,8,
208,235,165
2030 DATA 203,56,233,64,133,203,165,204,233,1,133,204,16
9,0,141,103
2040 DATA 6,169,0,162,7,30,104,6,106,202,16,249,141,255,
255,238
2050 DATA 77,6,208,3,238,78,6,238,103,6,173,103,6,201,8,
208
2060 DATA 224,200,192,40,208,178,96,0,0,0
2999 REM Build a new display list.
3000 DL=PEEK(560)+256*PEEK(561)
3010 POKE DL+3,78:POKE DL+99,78
3020 FOR I=6 TO 98:POKE DL+I,14:NEXT I
3030 FOR I=102 TO 198:POKE DL+I,14:NEXT I
3040 RETURN
```

Listing 7. BASIC Epson & Micropainter.

CRACKING THE CODE Part 2

By Keith Mayhew and Roy Smith

Since the last issue, you have had plenty of time to practice working with binary and hex numbers. This second part will discuss briefly the general internal layout of the machine and proceed to some introductory machine code instructions.

Basics

The microprocessor used in the ATARI computers is the popular and well documented 6502. The 6502 has sixteen address lines and eight data lines to communicate with the rest of the computer. In part 1 we showed a diagrammatic representation of binary codes in columns, and we showed up to eight columns or bits. If you can imagine another eight columns added to the first eight this would represent our total number of address lines. So the columns would represent:1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768. If every column contained a '1' then the largest number accessable would be the total of all sixteen columns, which would be 65535. If every column contained a '0' then the lowest number is obviously zero, which means the overall range of addresses is 65536. Now is a good time to introduce an abbreviation, that is the 'K' representing 1024. If 65536 is divided by 1024 then it can be written as '64K'. Each one of the 65536 addresses can be thought of as a box, into which a piece of data may be stored. The name given to one of these boxes is the 'byte', which is further sub-divided into eight bits (one bit is equivalent to a binary digit). As we know, eight bits can represent a number from 0 to 255, giving a total of 256 different numbers. Thus all 65536 memory locations have a number between 0 and 255 held in them, which is transferred via the eight data lines.

64K of available memory sounds like an awful lot! But that is the trouble, it is not all available to you for program and data storage. In the ATARI without BASIC, 48K of memory is left free for program storage in RAM (Random Access Memory). The rest of the 64K is allocated to the ROM (Read Only Memory), which is where the operating system is held. The general I/O devices are also allocated some of the memory, these devices include the cassette recorder, the display and the keyboard.

The 6502 Processor

The 6502 processor contains several different registers. All of these registers are eight bits wide except for the 'program counter', often denoted as 'PC', which is sixteen bits wide. The other registers are the 'accumulator' or 'A', the two index registers 'X' & 'Y', the processor flags register 'P', and the stack pointer 'S'. The PC register is sixteen bits wide because it needs to point to any byte in the full 64K memory space, the byte pointed to is the byte where information will be accessed. The most often used register is the accumulator, this is where most calculations and values are held. The two index registers are general purpose and can be used by themselves or in conjunction with the accumulator. The P register contains the current status of the processor and the S register points to the current position in the stack. This is only a brief introduction to the 6502's registers and these will be covered in greater depth later.

The Paging Concept

Due to the 'architecture' or internal layout of the 6502, the memory is set out in 'pages' consisting of 256 bytes per page, this conveniently divides into the 65536 locations to give 256 pages. Why are the memory locations laid out in this way? The reason lies with the fact that the 6502 is an eight bit microprocessor, and as the address bus is sixteen bits wide, it is composed of two eight bit words (where a word is an expression used to describe a set of bits of a certain length; note, an eight bit word is often referred to as a byte). The upper eight bit word points to one of 256 pages and the lower eight bit word points to a byte within that page, see figure 1.

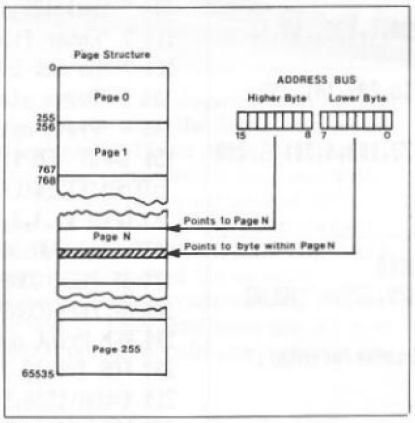


Figure 1

This paging feature is not generally of any importance, as the complete memory block appears to the programmer as one continuous area of memory locations. However, it is important to consider the paging feature when programming in machine code for two reasons, the least of which is a slight time delay when crossing a page boundary. The other reason is connected with limitations to some 'addressing' modes, where that mode is restricted to only one page in size. More on addressing modes later.

Machine Code Instructions

Machine code programs are stored in memory as a series of binary numbers. The 6502 starts execution by reading the first byte pointed to by the Program Counter register (PC), this will inform the 6502 which command it is to implement, and is referred to as the operation code or op-code byte. This op-code is followed by either none, one or two additional bytes depending on the type of instruction being executed. These bytes are automatically read by the 6502, incrementing the PC register by one each time. These bytes are often referred to as the operand bytes and their purpose is either to point to a memory address where the op-code will perform its function, or the operand will actually be the data which the op-code is to act upon. In other words in one instance the op-code needs the actual value of the operand byte as the data for its operation, whereas in the other case the op-code retrieves its data from the location pointed to by the operand byte(s). Unfortunately, unlike BASIC which reports errors to the user, the 6502 gets totally confused when it retrieves an op-code which is not one of its recognised instructions. This will cause the 6502 to 'hang up' or 'crash' and is irreversible apart from turning off and on and starting all over again!

The Hardware Stack

The 6502, as mentioned earlier maintains a Stack Pointer (S). The S register is eight bits wide and can therefore address anywhere within a page size. In the 6502, the S register addresses page one. Page zero is reserved because it has great importance to some of the 6502's instructions, therefore the stack, which could be on any page, has simply been allocated to the next available page. This stack is referred to as the hardware stack because it is supported by the 6502, other stacks do exist but are supported by software techniques, for example BASIC keeps a 'run time' stack to place the return address for a subroutine.

The stack is used for high speed, temporary storage of numbers, where it is impractical to use the limited number of internal registers. The internal registers A, X and Y are used for quick manipulation of numbers, but if you need to save some numbers temporarily whilst executing other routines, the stack is ideal. The only other way of saving these numbers would be to place them in RAM, this is slightly slower, but more importantly you need to keep a track of the memory location being used, whereas the stack being hardware maintained, keeps track for you. So, why not use the stack for all storage? There are two reasons, first is the stack is limited to one page of storage i.e. 256 bytes. The other reason is that the stack is a sequential storage system, which means that numbers can only be retrieved in the order in which they were stored. This type of stack is called a LIFO stack, standing for Last In First Out.

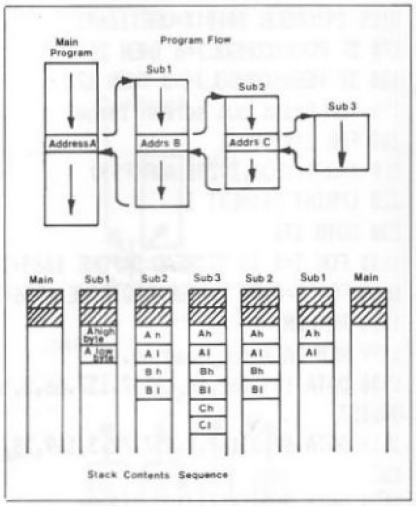


Figure 2

A particular use for the stack is subroutines. When the subroutine is called, the current contents of the PC register is stored on the stack, then it is loaded with the address of the subroutine which is then executed, upon completion the PC register is reloaded from the stack with the address of where it left off from the main program. It should be noted that if in a subroutine numbers are stored on the stack, they

obviously must be retrieved within that subroutine before returning to the main program, otherwise the return address will be incorrect, and will certainly cause undesired results. If subroutines are nested, i.e. one subroutine calls another, see figure 2, then care should be taken to ensure that not more than 256 bytes are stored on the stack or 128 subroutines maximum, otherwise information will be overwritten because the stack loops around on itself. In practice less than 128 subroutines can be called because there will probably be information already stored on the stack.

Addressing Modes

There are six main types of addressing modes: Immediate mode is when, as mentioned earlier, the operand of the instruction is the actual data required, see figure 3. Absolute mode is the other case mentioned earlier where the operands point to the location of the data required, see figure 4, note this mode covers the complete memory area including page-0. Short addressing or page-0 addressing, is the same as absolute except only one byte is needed to point anywhere in page-0, see figure 5. Remember, absolute addressing covers this page as well, it's just slower and needs a second byte. Indexed addressing uses an absolute base address and adds to it the contents of either the X or Y registers to give the final computed address, see figure 6. Indirect uses two locations in page-0 to point to another memory location, see figure 7. The last type is Implied mode which needs no data and just performs a set, internal function. These are only the basic addressing modes, and can be combined in many different ways, which will be covered in due course. These addressing techniques apply to all instructions, with a few exceptions. The main point to understand is that all of these different addressing modes are really only different ways of accessing data, with the exception of implied mode.

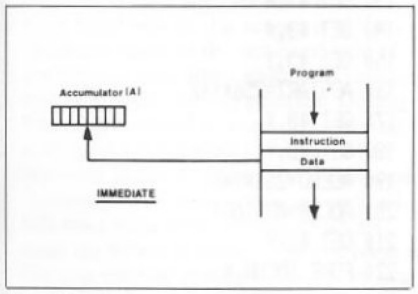


Figure 3

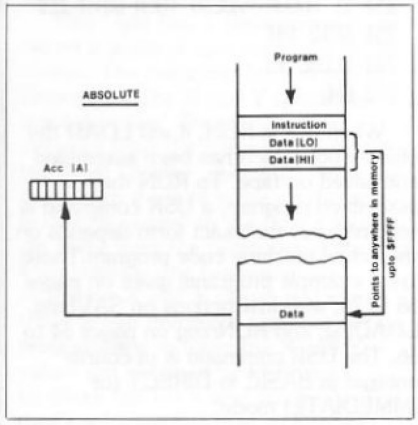


Figure 4

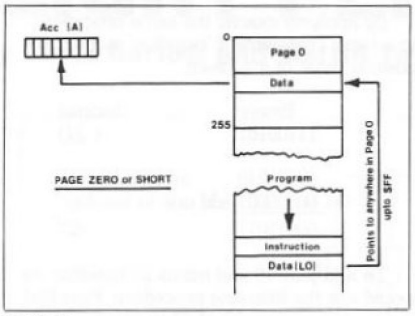


Figure 5

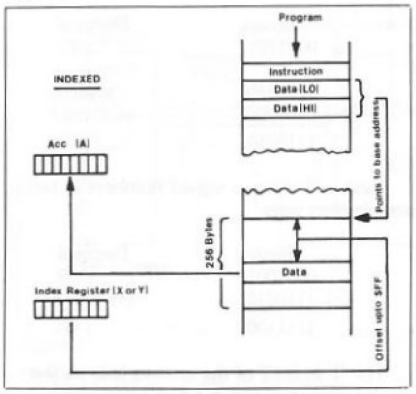


Figure 6

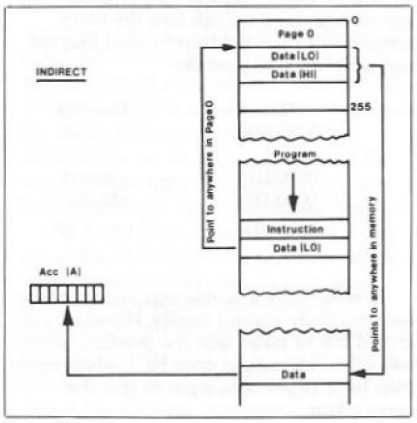


Figure 7

The Processor's Flags

Almost all of the instructions that can be used, affect the processor's flag register (P). Different instructions affect different bits of the register, where each bit will be either true (1) or false (0). What an instruction actually does will depend on the status of a particular bit in the flag register. Referring to figure 8, each bit has the following meaning:

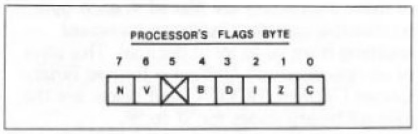


Figure 8

Bit 7, (N), a '1' indicates that a result was negative, a '0' indicates that a result was positive.

Bit 6, (V) a '1' indicates that a result had overflowed, a '0' indicates that it had not.

Bit 5 is not used in the P register.

Bit 4, (B) a '1' indicates that an interrupt was caused by the BRK instruction, a '0' if it was not.

Bit 3, (D), a '1' indicates decimal mode is in use, a '0' indicates binary mode is in use.

Bit 2, (I) a '1' indicates that interrupts are enabled or allowed, a '0' indicates that they are disabled.

Bit 1, (Z), a '1' indicates that the result of an operation was zero, a '0' indicates the result was non-zero.

Bit 0, (C), a '1' indicates that a carry has occurred, a '0' indicates that there is no carry.

Bits two and four, concerning interrupts will be covered in greater depth in a future article, but all the others lead us onto arithmetic operations.

Sums!

To finish, we have to get down to some mathematics, sorry, but it really is necessary!

(1) Unsigned Numbers: The numbers we have encountered so far have all been 'unsigned', this means that they have all been positive and we have no way of representing a negative number yet. To show the relevance of the carry (C) and overflow (V) flag we will demonstrate some binary arithmetic on eight bit numbers:

	Binary	Decimal
	00010101	21
(+)	00000111	+ 7
(=)	00011100	28

O.K., how did we go about adding up the binary numbers? Starting from the right hand column, we add 1+1 which equals 2. Two in binary is '10' so we put the zero in the answer line and carry the '1' into the next column. Now we have 1+0+1 which equals 2 again. So once again we put the zero in the answer line and carry the '1' into the third column. This column is now 1+1+1, which is 3. Three in binary is '11', so we put '1' in the answer column and carry the other '1' into the fourth column. The fourth column is 1+0+0 which is '1', this one is entered into the answer column and there is nothing to carry forward. The next column is just 1+0 which is a '1' in the answer column. The last three columns are all 0+0's which of course is '0'!

Let's do another example:

	Binary	Decimal
	10011000	152
(+)	10001110	+ 142
(=)	100100110	294

Following the same procedure as shown before, the answer comes out nine bits long due to the carry from the last column. As the 6502 only works in eight bits then any two eight bit numbers will need up to nine bits when they are added together, so the maximum would be:

	Binary	Decimal
	11111111	255
(+)	11111111	+ 255
(=)	111111110	510

The 6502 places this ninth bit or the 'carry' into the carry flag (C) of the register (P) i.e. sets the carry flag to a '1'. Note that the carry flag is left set until reset by the programmer.

(2) Signed Numbers: Numbers can be assigned a polarity of either positive or negative, the way this is done is to allocate

the most significant bit (bit 7) as a zero to represent positive or a one to represent negative. Here are some examples:

00001011 = 11, decimal.

The far left hand bit (msb) is a '0' which means the number will be positive, in this case +11.

01111111 = 127, decimal.

Again the number is positive, but using this method +127 is the highest number we can represent.

10001011 = -11, decimal.

The one in bit 7 represents negative, therefore the number is minus 11.

11111111 = -127, decimal.

Again this is the largest negative number which can be represented with this method. Adding two signed numbers can result in complications:

Binary	Decimal
01010111	+ 87
(+) 01110000	(+) +112
(=) 11000111	-71 Wrong

By carrying a '1' from bit 6 into bit 7, the sign has been accidentally changed to a minus number to give an incorrect result. When adding these numbers on the 6502, it has no way of knowing that they are actually signed numbers so it makes no adjustments for you, so it is up to the programmer to correct the result if this occurs. The 6502 does however give an indication that the sign bit has been changed by placing a '1' into the overflow flag (V) of the register (P). Therefore by clearing (V) before arithmetic and looking at (V) after, to see if an overflow has occured into bit 7, any necessary adjustments can be made. Using the signed method is not very reliable as there are many instances when the result will be incorrect even when no overflow is encountered:

Binary	Decimal
00010011	+19
(+) 10011011	(+) -27
(=) 10101110	-46 Wrong

Obviously, plus 19 added to minus 27 should equal minus 8 and not minus 46. It is interesting to note that the resulting answer is in the correct sign i.e. minus, and +19 added to plus 27 is plus 46, so you can see where this wrong answer came from!

(3) Two's Complement: Because of the problems shown above, there is a need for a better system for representing signed numbers. In 'two's complement' positive numbers are still represented by a zero in bit 7 and then the rest of the bits will be the value of the number up to plus 127. To invert the sign of any number i.e. plus to minus and minus to plus, the following functions are applied to the number: first each bit is inverted i.e. '0' becomes '1' and '1' becomes '0', and then 00000001 is added to the resultant number.

To represent minus 27, we first write the binary code for plus 27, then we invert each bit, and add one to give the two's complement *representation* of minus 27.

Binary	Decimal
00011011	+27
11100100	invert each bit.
(+) 00000001	add one to number.
11100101	(-27)

By applying exactly the same procedure to a two's complement 'negative' number the positive value of it is given.

Binary	Decimal
11100101	(-27)
00011010	invert each bit.
(+) 00000001	add one to number.
00011011	+27

To add plus 19 and minus 27 together we would use the following procedure: First find the two's complement of 27 to give minus 27:

Binary	Decimal
00011011	+27
11100100	invert.
(+) 00000001	add one.
11100101	(-27)

Next add the two signed numbers in two's complement form;

Binary	Decimal
00010011	+19
(+) 11100101	(+) (-27)
11111000	(-8)

The '1' in bit 7 of the answer tells us the result is negative and is a two's complement representation of minus 8. Just to show that this represents -8 we can take the two's complement again but bear in mind that the sign will change to positive:

Binary	Decimal
11111000	(-8)
00000111	invert.
(+) 00000001	add one.
00001000	+8

Now we have a usable sign system where we can ensure correct results. However, care should still be taken with the overflow which will still accidentally change bit 7, which must then be changed back again to give the correct sign.

(4) Decimal mode: To display an eight bit binary number on the screen is reasonably easy in hex format, as two four bit blocks can be considered, each representing a hex character. But to display this on the screen in decimal characters is comparatively difficult. For this reason the 6502 has a 'decimal' mode, which uses two four bit blocks to represent two decimal characters. The codes of 0 to 9 are all that are needed and so 10 to 15 are unused, two of these characters are placed in each byte so that the complete byte can represent anything from 00 to 99 in decimal. This form of storing decimal numbers is termed Binary Coded Decimal (BCD). Listed below are the four bit binary codes for '0' to '9'.

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Using this information, let's try some examples in eight bits:

BCD	Decimal
0101 0001	
5 & 1	= 51
1001 0010	
9 & 2	= 92
0110 1000	
6 & 8	= 68

In this decimal mode the 6502, when adding numbers, automatically does an internal carry between bits four and five, if the number exceeds the decimal value for that block i.e. 9. This mode is activated when the (D) flag is set to one in the (P) register.

This completes most of the basic groundwork needed for a thorough understanding of machine code, next time we will be covering assembly language programming in some depth.

DID YOU KNOW

Object Lesson

The instructions on page 23 of the ATARI Assembler Editor manual, on loading OBJECT code to cassette in EDIT MODE are incorrect. Note that the SAVE procedure on page 64, i.e. ASM,,#C: does work and will save the assembled (OBJECT) code to tape. The loading problem can be overcome by inserting the BASIC cartridge and keying in the following program:

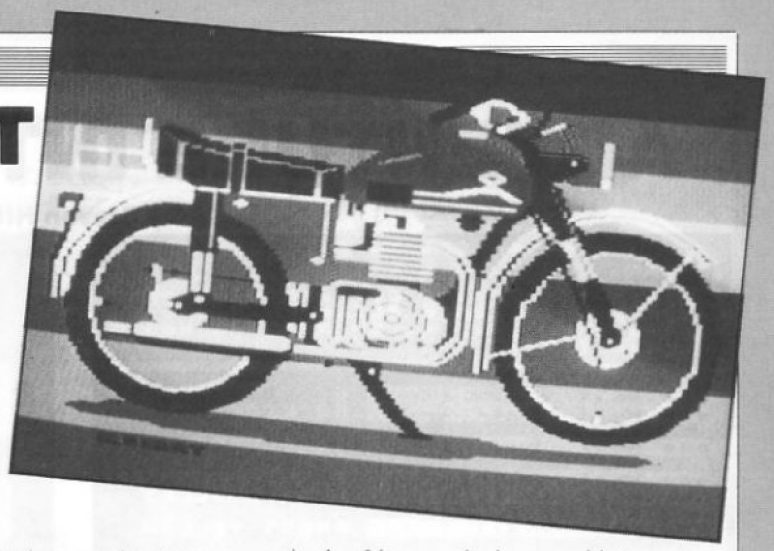
```
100 TRAP 260
110 OPEN #3,4,0,"C:"
120 GET #3,X
130 GET #3,X
140 GET #3,X
150 GET #3,Y
160 ADSTART=256*Y+X
170 GET #3,X
180 GET #3,Y
190 ADEND=256*Y+X
200 ADCUR=ADSTART
210 GET #3,X
220 POKE ADCUR,X
230 ADCUR=ADCUR+1
240 IF ADCUR<=ADEND THEN GOTO 210
250 GOTO 140
260 CLOSE #3
270 END
```

When this is RUN, it will LOAD the object code which has been assembled and saved on tape. To RUN the assembled program, a USR command is required, but the exact form depends on the actual machine code program.There are 4 example programs given on pages 68 to 74, with instructions on SAVEing, LOADing, and RUNning on pages 64 to 66. The USR command is of course entered in BASIC in DIRECT (or IMMEDIATE) mode.

E. O. Rice - Stanmore.

FUN WITH ART

By Keith Berry — Birmingham



The superb graphics of Atari computers has encouraged a steady flow of programs enabling the user to indulge in "Computer Art", drawing pictures directly on the screen sometimes with the aid of Paddles, but mostly using a Joystick. The culmination of this type of programming was Reston's "PAINT" diskette, which has recently been officially adopted by Atari for distribution as their own.

Two recent deviations from the Joystick method are the program supplied with the Atari Touch Tablet and "PAINTBOX", which is about to appear from Liverpool for use with the PIXSTICK Light Pen. One of the latest additions to this genre of programs is "FUN WITH ART" from EPYX. It costs just under £40 and uses a Joystick, but it has three additional features to make it stand out from those which have gone before:

1) It is in the form of a ROM cartridge, which is great news for the cassette user with at least 32k RAM.

2) Although "painting" is executed with a choice of only four colours at a time, including the background, the program can set up its own Display List Interrupts, so that changes in hue or luminance mean that up to any three of the colours at a time can be changed in horizontal bands almost anywhere on the screen.

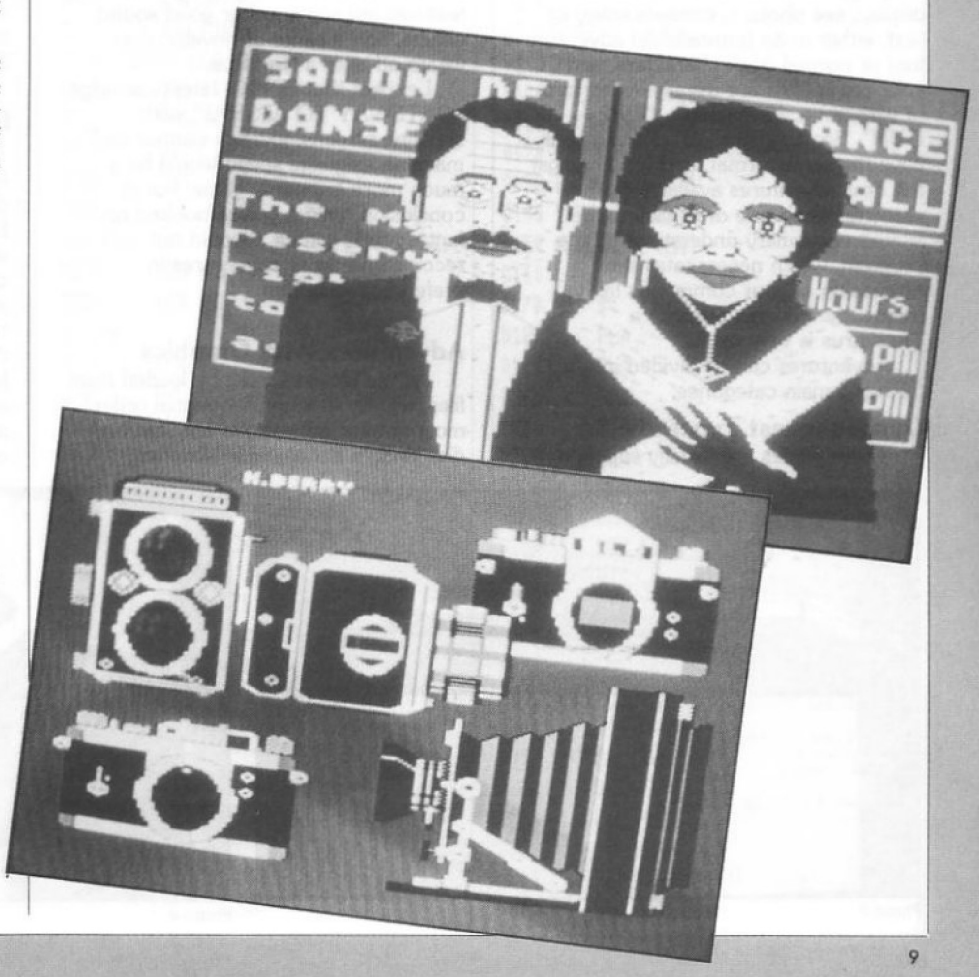
3) It is supplied complete with the listing of a BASIC program, to enable pictures that have been saved to tape or diskette can be displayed independently of the cartridge, including all of the D.L.I's mentioned above.

Saving and Loading a picture takes time though. A C60 cassette stores about five pictures per side, and on diskette each screenful uses 65 Sectors. During the painting operation you switch back and forth at the touch of the START key between the picture that you are creating and a colourful Menu, the options of which are selected either by moving a cursor with the Joystick, or by a single keypress. Using the latter method, there is no need to leave the picture screen to change painting mode once you have learnt or noted down the codes, which are shown in inverse at the foot of the menu. Although it is quite possible to create a satisfactory image by plunging directly into the blank screen, making the most of the D.L.I. colour changes requires some fairly careful advance planning.

There are too many features to review individually, but here are the main ones: three "brush" widths, two sizes of text (as Gr.1 & 2), plot a single pixel, draw a single line, a continuous outline or plot a circle. Any size block of pixels can be moved, copied, inverted, reversed, deleted or saved to tape or disk independently of the rest of the picture. Any small part of the picture can be magnified to fill the screen, for precise attention to individual pixels. The occurrence of a particular colour in a specified area can be changed instantly to any other colour. Colours can be given priority over (or under) the others in four priority steps, one can draw a line "behind" an object of one colour and "in front" of an object of another. For disk users, a directory of the diskette can be viewed through the text window without leaving the program.

This is one of the most user-friendly programs that I have seen, which is just as well since I lost sight of the instructions after a single reading! There is only one feature, Block Colorform, that I have so far been unable to use without the aid of the instructions, so if anyone could kindly let me have a copy of the relevant paragraph I would be most grateful.

If you have the August 1983 issue of ANTIC Magazine, have a look at Peter Wickman's "EARTH" on page 33, a superb picture that has been taken up by EPYX in their advertisements for the cartridge. My overall view of "FUN WITH ART" is that it provides as much enjoyment as any arcade game, with the bonus of an "end-product". It reveals the superiority of the Atari over most other computers, by allowing the use of the full range of colours and luminances with only minor restrictions. I can recommend it without reservation to anyone with an inclination towards Computer Art.



ADVENTURE INTO THE ATARI

By Steven Hillen

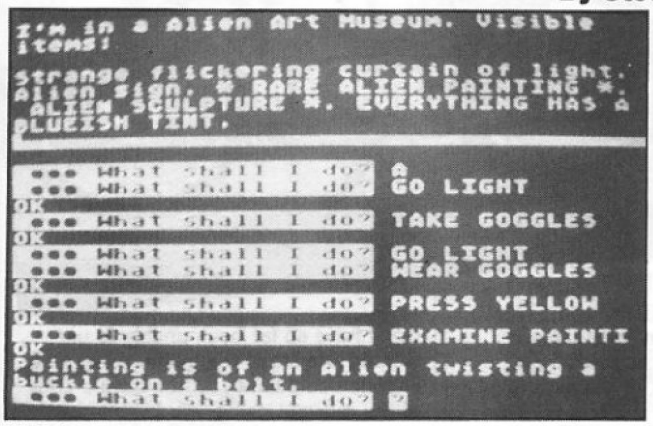


Photo 1

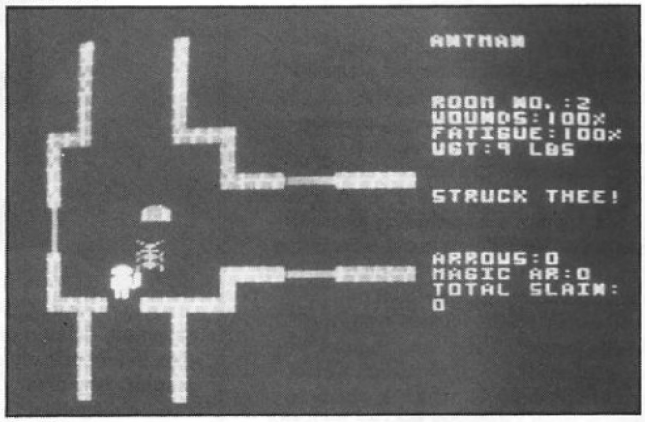


Photo 2

Imagine you are on a small scouting ship, marooned on a tiny planet at the edge of the galaxy. Behind the jammed maintenance hatch, is the damaged fuel rod which must be replaced if you are to escape from the planet. Armed with a gun, and protected by a space-suit with only a limited amount of air, you set off to explore the planet, unearthing alien technologies, priceless treasures, and with luck, a new fuel rod!

As you may recognise, the description above is from "STRANGE ODYSSEY" by Scott Adams (24K tape) and is a good example of the 'Text-Only' type of adventure. The screen display, see photo 1, consists solely of text, either in an (unreadable) adventure font or normal Atari characters, and your commands are usually in the form of verb-noun, for example, "FIRE GUN". Scott Adams' 12 adventures all have the same format, and are amongst the best adventures available on cassette. Their one drawback is the limited vocabulary understood by the program, which necessitates the rephrasing of your commands fairly frequently. For this purpose a Thesaurus is very useful!

Adventures can be divided into a further 3 main categories:

Dungeonquest Types

These games are usually supplied with books of lore that are often of better quality than the programs themselves. No thinking is involved in these adventures, you simply guide a character around a block-drawn map, fending off giant spiders, and grabbing various treasure caskets. A typical example is "TEMPLE OF APSHAI", see photo 2. Believe me, it is not particularly exciting to type "F" and watch a flickery arrow move across the screen and destroy a swarm of killer bees. These games play very slowly, all commands being typed on the keyboard, with BASIC taking up to 10 seconds to redraw the screen. Their only redeeming features are some rather good sound effects, and a sense of novelty, that wears off after a few days.

I expect that EPYX's latest cartridge "GATEWAY TO APSHAI", with scrolling screens, joystick control and machine language speed would be a much more enjoyable game, but in conclusion, unless you're hooked on fantasy type games, I would not recommend these adventures in preference to others.

Adventures With Graphics

As the pictures must be loaded from files, usually in a non-sequential order, most graphic adventures are sold on disk only. It is debatable whether graphics enhance an adventure or not. Certainly with Scott Adams Graphic Adventures, the amount of flicker generated by the display is quite unacceptable, and detracts from otherwise good adventures.

At the top of the range however, is Datasoft's "SANDS OF EGYPT" (16K disk only, see phote 3) which has some of the most colourful graphics I have seen on any adventure program. The screen is split into a text region at the bottom where commands are entered, and a graphics region above, which shows your current position in a scrolling 3D view. This is one of the more difficult adventures, where you must do things in exactly the right order if you are to survive.

Infocom Adventures

I have saved the best until last! Infocom adventures (ZORK I, II, III; STARCROSS; DEADLINE; etc.) all have a very large vocabulary (over 600 words), and commands are typed in as complete sentences, for example, "Drop all but the sword, take the coal and open the machine". This avoids most of the tedious rephrasing necessary with less sophisticated adventures. The size and complexity of these text only adventures, are far greater than any others, and the descriptions of the

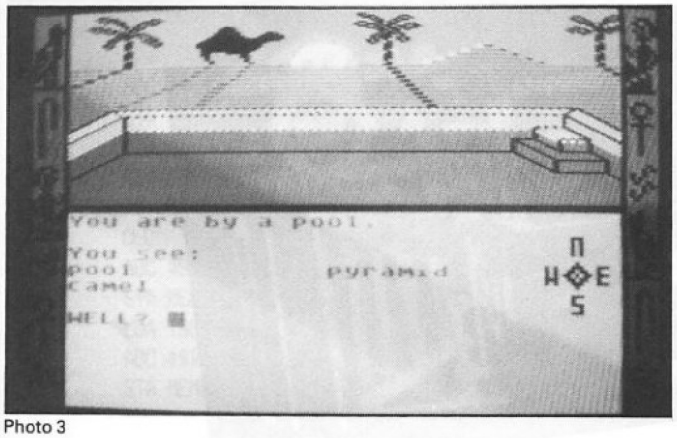


Photo 3

Photo 4

locations often take up more than one screen. The attention to detail is meticulous, and the amount of interaction between adventurer and objects or characters within the program makes for a plausible game. The packaging of these games is as excellent as the programs themselves. STARCROSS, for example, is supplied in a white saucer-shaped spaceship, with a starmap and comprehensive space-mining manual, see photo 4.

Fundamentals

Even if you have not been adventuring yet, you will now know about the main types available. For those of you who are new to adventuring, there are two basic rules to obey, which can save hours and hours of frustration.

The first is that the game position should be saved to cassette or disk just before you attempt anything dangerous. Thus, if your hazardous manoeuvre ends in disaster, then you can merely reload your saved position and try a less dangerous solution. This saves tiresome re-typing of all the commands to reach that position again.

Secondly, it is a wise move to draw a map. Probably the best method to use is to represent each location by a place name in a box. As you arrive at a new location, all the exits can be noted in that box, and arrows, labelled with suitable directions, can be used to connect the boxes (see figure 1). If the

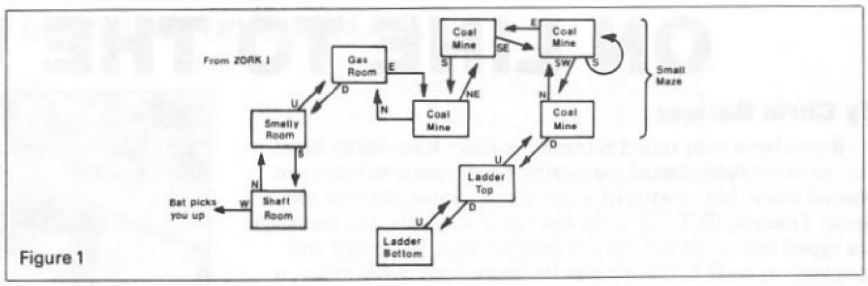


Figure 1

physical shape of the world you are exploring is odd, the map may be a little tricky to draw, e.g. in Starcross you are on the inner surface of a giant cylinder. The only other problem in drawing maps is when you encounter a maze, where all the location names are the same for different rooms. A "Hansel and Gretel" technique should be adopted, you drop objects in the maze enabling you to differentiate between separate rooms. These mazes are perhaps the most laborious parts of adventures to be solved, but most games seem to include them.

Over To You

This is the first of a (hopefully) regular column, and I intend that it should be a mixture of reviews and coded hints for adventurers. The success of this column will depend largely upon you, if you've been trying to get past a gremlin for several months, but still can't find a solution, then you should write to me at the Club address, giving exact details of the problem. If my meagre talents cannot answer your query, then I shall print it in the next issue and maybe one of you will be able to reply. One thing though, please do not ask for hints or solutions without giving the adventure a good go first, as it may well spoil your overall enjoyment and satisfaction at "beating the program".

I would also be pleased to hear from you about any particular adventures that you would like to see reviewed. Finally, I shall start the ball rolling by asking these two questions:

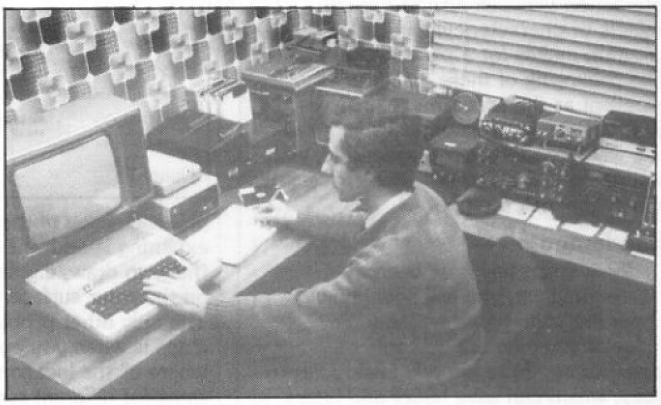
(A) How do you remove the blue rod from the silver sphere, without using the gun in "Starcross"?

(B) How do you get the iron glove in "Pyramid of Doom"?

These two problems have been annoying me for some time and if I receive any answers, I will print coded solutions with the adventurer's name in the next issue.

ON LINE TO THE WORLD

By Chris Barlow



If you have ever tuned around the short wave radio bands you have probably heard many strange signals which are not normal voice, but consist of audio tones. Some of these are Radio Teletype (R.T.T.Y.). As the name suggests, the messages are typed out on an ordinary typewriter style keyboard and received on an R.T.T.Y. printer or, more commonly today, a video display unit (V.D.U.). This method of sending messages has been around for about as long as radio, but the code used dates back even further.

Invented by J.M.E. Baudot in 1860 and although modified by D. Murray in 1903, it is still referred to as the Baudot Code. The code is made up of five bits of data and two stop bits. This gives 32 possible combinations, enough for all 26 letters of the alphabet, plus control characters for line feed, carriage return and space bar. There are two special control codes, one is called 'Letters' the other 'Figures'. When the Letters control code is sent all the following data will be interpreted as letters of the alphabet until the Figures control code is received. This will then interpret the 26 codes as numbers, 0 to 9 and most punctuation marks, as can be seen in Fig. 1.

In R.T.T.Y. there are two audio tones used, the Mark Tone and the Space Tone. When received, the Mark Tone gives the logic low state and the Space Tone gives the logic high state. This output is then fed to the R.T.T.Y. printer, or V.D.U. In audio frequency shift keying (A.F.S.K.), an amplitude or frequency modulated carrier is transmitted. On the short wave bands both radio amateurs and commercial stations use, for the most part, frequency shift keying (F.S.K.). This is where the transmitted carrier frequency is made to shift by a controlled amount when the logic state changes from low to high. To obtain the two tones for decoding purposes, it is necessary to have a beat frequency oscillator (B.F.O.), or a single sideband (S.S.B.) position on your receiver. This method calls for more accurate tuning to obtain the correct tones for decoding by the terminal unit (T.U.). On the V.H.F. and U.H.F. bands, radio amateurs also use A.F.S.K., so tuning is not so critical.

The T.U. in its most simple form is no more than two audio frequency filters. However, these filters must have a narrow response so only the relevant tones can pass. The output of the filters are then converted into digital signals giving the low and high outputs required by the R.T.T.Y. printer or V.D.U. To obtain a useable piece of hardware is a little more complicated and I have written an article for the Maplin Electronics Magazine showing a circuit diagram and construction details. A kit of parts will soon be available from Maplin Electronics, the cost of which is yet to be announced. The article is due to be published in the September 84 issue of ELECTRONICS (the Maplin Magazine), under the name 'TU1000'.

The output from the Maplin TU 1000 is at RS 232 levels. To decode this you will need the Atari 850 Interface Module and the software to translate the Baudot to ASCII so that the ATARI Computer can display the messages on the screen. You may be familiar with telephone modems using 300 Baud to transmit and receive data. R.T.T.Y. messages are usually sent at much slower speeds. Radio amateurs use 45.5 or 50 Bauds although commercial stations can use higher Baud rates. The 850 Interface Module can handle, in the 5 bit mode, 45.5 to 300 Baud. This should prove more than adequate for most stations received.

In the following program, the setting-up of the first RS 232 port for concurrent 5 bit input mode is achieved by using XIO commands to the 850. Unfortunately, when in less than eight bit mode, the 850 can only send out data in its block mode. This means the buffer inside the 850 stores 32 characters and then transmits them in one block. However, if a carriage return is sent the block of data is then transmitted at any time. When in receive mode, by pressing keys 1 to 8, the Baud rate is set from 45.5 to 300 Baud, and is displayed. By pressing the option key, the transmit facility is activated and will stay in this mode until the start key is pressed after your message has been transmitted.

	Bit					Decimal Value
	5	4	3	2	1	
LETTERS	1	1	1	1	1	31
FIGURES	1	1	0	1	1	27
LINE FEED	0	0	0	1	0	2
CARRIAGE RETURN	0	1	0	0	0	8
SPACE BAR	0	0	1	0	0	4

Letters	Figures						
A	–	0	0	0	1	1	3
B	?	1	1	0	0	1	25
C	:	0	1	1	1	0	14
D	$	0	1	0	0	1	9
E	3	0	0	0	0	1	1
F	!	0	1	1	0	1	13
G	& or +	1	1	0	1	0	26
H	#	1	0	1	0	0	20
I	8	0	0	1	1	0	6
J	'	0	1	0	1	1	11
K	(	0	1	1	1	1	15
L	)	1	0	0	1	0	18
M	.	1	1	1	0	0	28
N	,	0	1	1	0	0	12
O	9	1	1	0	0	0	24
P	0	1	0	1	1	0	22
Q	1	1	0	1	1	1	23
R	4	0	1	0	1	0	10
S	BELL	0	0	1	0	1	5
T	5	1	0	0	0	0	16
U	7	0	0	1	1	1	7
V	;	1	1	1	1	0	30
W	2	1	0	0	1	1	19
X	/	1	1	1	0	1	29
Y	6	1	0	1	0	1	21
Z	"	1	0	0	0	1	17

Figure 1

The program is still in its infancy and I am sure you could make many useful additions to those offered. I would be very interested to see any rewritten versions.

The number and quality of the stations received will depend upon the performance of your receiver and the efficiency of your aerial system. Most modern communication receivers will be adequate, the cost of which start at £150 and upwards! There are still many ex-commercial receivers available at quite reasonable prices which offer excellent performance. Another consideration is the amount of interference generated by the rest of the hardware. The Atari Computer has a very low radio noise output and is unlikely to present any problems. This is because Atari have to meet the American standards. Domestic Televisions are not built to such high standards, therefore they do radiate a lot of unwanted noise which is picked up and amplified by the receiver. This will swamp weak signals and generally interfere with your pleasure. This problem has to be accepted, but you can still receive a vast number of interesting signals from around the world. I have personally received transmissions from N. America, S. America, Africa and Asia.

The variety of the messages received are varied. Radio amateurs chat between themselves, whilst commercial stations pass weather information, news, telegrams, Stock Exchange, Financial Markets, Aircraft Flight plans, etc. Some of these messages are in code and you could spend many hours trying to decipher them. This small package of hardware and software could lead to many hours of enjoyment and the possibility of yet another interest.

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

```
20 REM 850 INTERFACE MODULE RS-232 PORT 1.
30 TRAP 720:CLOSE #2:OPEN #2,4,0,"K:"
40 GRAPHICS 0:POKE 710,180:POKE 709,1:POKE 712,180:POKE
752,1
45 POKE 16,64:POKE 53774,64
50 PRINT "  RYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYR"
60 PRINT "  Y                                Y"
70 PRINT "  R GBLVI. R.T.T.Y. SEND/RECEIVE. R"
80 PRINT "  Y                                Y"
90 PRINT "  RYRYRYRYRYRYRYRYRYRYRYRYRYRYRYRYR"
100 SOUND 0,80,10,5
110 FOR LOOP=0 TO 17:READ D:POKE 1536+LOOP,D:NEXT LOOP
120 MAC=USR(1536)
130 REM **** BAUDOT TO ATASCII SET-UP ****"
140 DIM BA$(7):DIM ATASCII(64):DIM BAUDOT(128)
150 FOR I=1 TO 64:READ DAT:LET ATASCII(I)=DAT:NEXT I
160 REM **** ATASCII TO BAUDOT SET-UP ****"
170 FOR I=1 TO 128:READ DAT:LET BAUDOT(I)=DAT:NEXT I
180 SOUND 0,50,10,5:PRINT "                READY ":POKE
752,0
190 FOR T=1 TO 500:NEXT T:KEY=49:SOUND 0,0,0,0:GOSUB 570
200 REM *** RECEIVE ***
210 LET SHIFT=0
220 STATUS #1,A
230 IF PEEK(53279)=3 THEN GOSUB 330
240 IF PEEK(764)<255 THEN GOSUB 550
250 IF PEEK(747)=0 THEN 220
260 GET #1,RECEIVE:GOSUB 280
270 GOTO 220
280 LET RECEIVE=ATASCII(RECEIVE-224+SHIFT+1)
290 IF RECEIVE<0 THEN RETURN
300 IF RECEIVE=0 THEN SHIFT=0:RETURN
310 IF RECEIVE=1 THEN SHIFT=32:RETURN
320 PRINT CHR$(RECEIVE);:RETURN
330 REM **** SEND ****
340 PRINT :PRINT "* SEND *  BAUD RATE ";BA$;:PRINT
350 CLOSE #1:OPEN #1,8,0,"R1:":LET AL=1
360 IF PEEK(53279)=6 THEN 530:REM START KEY TO RETURN TO
 RECEIVE.
370 IF PEEK(764)=255 THEN 360
380 GET #2,KEY:LET OUT=KEY
390 IF OUT>127 THEN LET OUT=OUT-128
400 LET OUT=BAUDOT(OUT+1)
410 IF OUT=0 THEN 360
420 PRINT CHR$(KEY);
430 IF AL=1 THEN 470
440 IF OUT<0 THEN 490
450 LET AL=1:PUT #1,31
460 GOTO 490
470 IF OUT>0 THEN 490
480 LET AL=0:PUT #1,27
490 PUT #1,ABS(OUT)
500 IF OUT<>8 THEN 360
510 PUT #1,2:PUT #1,31
520 XIO 32,#1,0,0,"R1:":LET AL=1:GOTO 360
530 CLOSE #1:OPEN #1,5,0,"R1:":XIO 40,#1,0,0,"R1:"
540 PRINT :PRINT " RECEIVE. BAUD RATE ";BA$:PRINT :RETURN
550 REM ..... BAUD RATE 45.5 TO 300...
560 GET #2,KEY:IF KEY>127 THEN KEY=KEY-128
565 IF KEY<49 OR KEY>56 THEN 660
570 IF KEY=49 THEN BA$="45.5 "
580 IF KEY=50 THEN BA$="50.0 "
590 IF KEY=51 THEN BA$="56.875 "
600 IF KEY=52 THEN BA$="75.0 "
610 IF KEY=53 THEN BA$="110.0 "
620 IF KEY=54 THEN BA$="134.4 "
630 IF KEY=55 THEN BA$="150.0 "
640 IF KEY=56 THEN BA$="300.0 "
650 RATE=KEY-48
660 BAUD=176+RATE
670 CLOSE #1:OPEN #1,5,0,"R1:":REM PORT 1 INPUT ONLY (CO
NCURRENT MODE)
680 XIO 36,#1,BAUD,0,"R1:"
690 XIO 40,#1,0,0,"R1:":REM PORT 1 CONCURRENT MODE SET
700 XIO 38,#1,32,0,"R1:":REM NO TRANSLATION
710 PRINT :PRINT " RECEIVE. BAUD RATE ";BA$:PRINT :RETURN
720 SOUND 0,25,10,10
730 POKE 752,1:PRINT :PRINT "< SYSTEM ERROR > ";
740 LET E=PEEK(195):PRINT E:FOR I=0 TO 300:NEXT I:RUN
750 DATA 104,169,6,162,6,160,11,32,92
760 DATA 228,96,169,0,133,77,76,95,228
770 REM BAUDOT TO ATASCII DATA.
780 DATA -1,69,-1,65,32,83,73,85,155,68,82,74,78,70,67,7
5,84,90,76,87,72,89,80,81,79,66,71,1,77,88,86,0
790 DATA -1,53,-1,45,32,144,56,55,155,36,52,39,44,33,58,
40,53,34,41,50,35,54,48,49,57,63,43,1,46,47,39,0
800 REM ATASCII TO BAUDOT DATA.
810 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,3,0,0,0,0
820 DATA 4,-45,-17,-20,-9,0,-26,-11,-15,-18,0,-24,-12,-3
,-28,-29,-22,-23,-19,-1,-10,-16,-21,-7
830 DATA -6,-24,-14,-30,0,0,0,-25,0,3,25,14,9,1,45,26,20
,6,11,15,18,28,12,24,22,23,10,5,16,7,30,19
840 DATA 29,21,17,0,0,5,0,0,0,3,25,14,9,1,45,26,20,6,11,
15,18,28,12,24
850 DATA 22,23,10,5,16,7,30,19,29,21,17,0,0,5,0,0
860 END
```

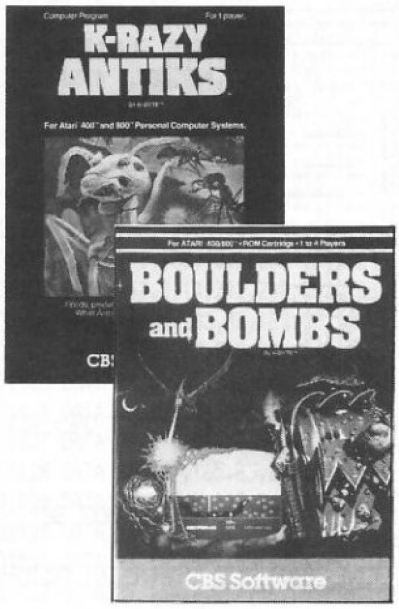
SPECIAL OFFERS

This quarter we present four more exciting software titles at very low prices for you to buy on special offer. A limited number of each title is available, so send in now, whilst stocks last.

KRAZY ANTIKS

Bent on the total extermination of the White Ants, hostile Enemy Ants are invading your anthill. The Enemy ants are relentless in their pursuit, your mission is to destroy them before they get you. Deposit White Eggs in the maze of tunnels to ensure survival of the colony. Destroy Enemy Eggs and lure the attackers into the deadly flood waters or into the path of the dreaded Anteater's tongue! Your goal is to skillfully manoeuvre all four Enemy Ants into the ant Trap, then you can advance to the next level. In this single player game you can select six different Anthills, which appear in the form of an intricate maze.

Cartridge for 400, 800, and XL range.
RRP £29.95 **Club Price £7.95**

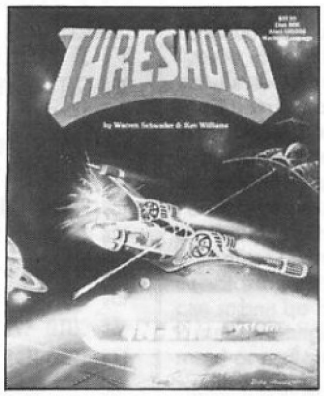


BOULDERS AND BOMBS

In this excellent game you must tunnel through the alien underground, blasting through boulders, eluding spreading green fungus and dodging nuclear rods which have been released by the Probe Birds. If you don't, you will be turned to fossil! You are trapped and your only means of escape is to dig a tunnel with your rotating cutter, but watch out for those boulders as they will cost you vital seconds. Time is vital, as the sun moves across the sky, sets and then the moon rises. You have only one day and one night to make your escape. As you forge ahead, pick up every bomb you pass, you will need them to blast your way out of tight spots! Once you successfully negotiate a quadrant you move on to the next, even more difficult level, and there are only 99 of them!

Up to 4 players can enjoy this entertaining game (only 2 on the XL computers, which is available on cartridge from CBS Software.

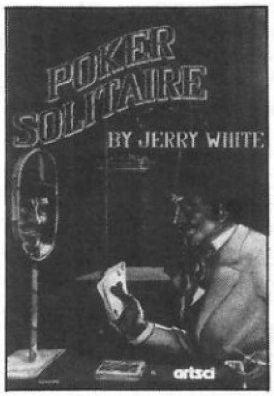
Runs on 400, 800 and XL range.
RRP £29.95 **Club Price £7.95**



THRESHOLD

If you love those "blast them" alien space games, then your collection is sadly lacking without a copy of Threshold. Not only is it fast and furious, this game loads new aliens for you to fight after each group has been defeated. To get to successive levels is no easy task, you'll need lightning fast reactions and cool nerves under fire. You only have five ships to start, but you will be awarded a bonus ship when you reach 50,000 points and every 100,000 points thereafter. You have to watch your laser guns do not overheat, if they do, you can only dodge the enemy until they cool down. It's claimed in the booklet supplied with the game, that the last group of aliens will be very special and are very easy to identify because of their characteristics. Well, as we have only ever managed to get to the third level, we cannot confirm this. Threshold is supplied on a diskette from ON-LINE Systems.

Diskette for 400, 800 and XL range, minimum 48K.
RRP £27.54 **Club Price £7.95.**



POKER SOLITAIRE

Try this variation of Poker, it will provide you with a perfect opportunity to hone your card playing strategies. Up to four players can test their abilities as they receive 25 cards which they place in a five by five array. The object is to place these cards to achieve the ten best poker hands possible. Sound easy? Sure does, but wait until you have to do it. Jerry White's Poker Solitaire will give card game addicts hours of entertainment. This program is available on cassette or diskette and only requires 16K. Please state if you require cassette or diskette when ordering.

Runs on 400, 800 and XL range.
RRP£14.95 **Club Price £3.95.**

READING MATTER

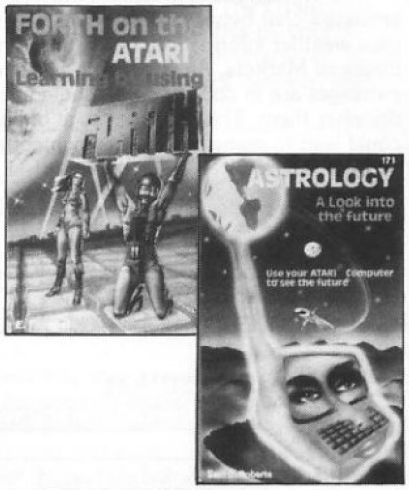
The following 2 books and Chart are also available from the Club, but stocks are limited so please order promptly to avoid disappointment.

FORTH ON THE ATARI

by E. Floegel.

Forth is an incredibly fast compiling language which, although more easily understood and used than machine code, does require considerable effort by the novice to obtain a full understanding of its capabilities. This excellent book not only explains Forth in terms the novice will understand, but also gives practical examples and hints which the experienced Forth user will find of great value.

RRP £6.50 **Club Price £3.95**

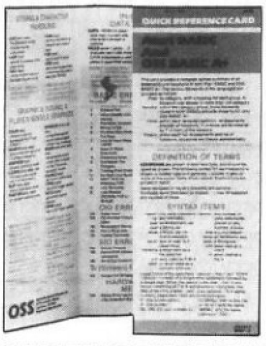


ASTROLOGY

by Sam D. Roberts.

A must for all Astrology buffs, this book is designed to help with the complex mathematics associated with Horoscopes and the reading of the Stars. Although a full program listing is supplied, reference is made to a Data disk which we are unable to supply, nevertheless this book is still full of useful information.

RRP £7.95 **Club Price £1.95**



QUICK REFERENCE CARD

This handy card unfolds to reveal lists of both ATARI BASIC and OSS BASIC A+ commands and functions. Ideal for both the novice and the experienced but forgetful programmer.

RRP £1.30 **Club Price 50p**

USER GROUP SOFTWARE

Software Librarian - Roy Smith

Due to demand from members there are now two ways to get programs from the library. The original scheme of exchanging '3 for 1' will still apply, but now with an added bonus. So the library rules have been extended to enable those members who cannot write their own programs to gain access, and those that can to have a possibility of some reward for their efforts. The extended library rules are as follows:

3 FOR 1 EXCHANGE

1. Every program you donate to the library entitles you to three programs in return.
2. The program you donate must be your original and not copied.
3. Your donated program must be submitted on a cassette or a disk, programs in the form of print-outs will not be processed.
4. If your program requires any special instructions they should be added in the form of REM statements within the program (or you may present them as instructions when the program is actually run).
5. BONUS. Every program donated per quarter (between issues of the newsletter) will be eligible to be judged 'STAR PROGRAM' for that quarter. This carries a prize of £10 which will be paid to the author from the club funds. The programs will be judged by the Editorial Team and their decision will be final. The Editorial Team are not eligible for the prize.
6. The '3 FOR 1' exchange is only open to club members.

DONATION SCHEME

1. Every club member will be entitled to ask for up to 3 programs per quarter from the library by donating to the club funds.
2. If a member does not take his/her entitlement for a particular quarter, it cannot be carried forward to the next quarter.
3. A member can have more than one quarter's entitlement at one time (up to a maximum of 12 programs (1 year)), but then will be unable to ask for more until his/her credit quarters have been used. Note that odd numbers of programs will be counted in quarters, i.e. if a member asks for 5 programs, the first 3 will be that quarter's entitlement, the next 2 will be the second quarter's entitlement and he/she will have to wait until the third quarter before he/she is entitiled to any more. Also note that having programs in advance will only be allowed if that member's membership covers the advance quarters.
4. The donation fee will be £1 per program and is not refundable. Cheques and Postal Orders are to be made out to the 'U.K. Atari Computer Owners Club'.
5. Members must send in a blank cassette or diskette for the chosen programs to be recorded on.
6. The 'DONATION SCHEME' is only open to club members.

Finally I would like to point out that some people omit to include return postage when donating to the library, so please do not forget to include 30p worth of stamps to cover this.

THE LIBRARY SOFTWARE SERVICE IS FOR MEMBERS ONLY

LIBRARY SOFTWARE TITLES

Listed below are software titles available to members under the "3 for 1" and donation schemes. 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 sent in a program of yours for exchange or donate to the club funds.

Games

BERTIE BLOCKHEAD
by Alex Kells - Liverpool.
Guide Bertie around the screen picking up the energy pills without receiving a lethal dose of radiation.
Runs in 16K Cassette or Disk min.

DRAGONFIRE
by G. Fairall - Oxon.
First you must get across the castle drawbridge before you can enter the treasure rooms, then you must avoid the Dragon's Fire.
Runs in 16K Cassette or 32K Disk min.

BALLOON LANDER
by David Campbell - Maldon.
Land your balloon on the landing pads. You must avoid low flying aircraft and take account of fuel, wind-speed & direction.
Runs in 16K Cassette or 32K Disk min.

GIL-BERT
by Mark & Brian Christian - Wirral.
Bounce on the squares to gain points but avoid the SKULLS.
Runs in 32K Cassette or Disk min.

TOWER
by Stephen Taylor - London.
Climb the Towers avoiding the falling buckets and pot plants.
Runs in 32K Cassette or Disk min.

CLOSE ENCOUNTERS
by Jeff Davis - Hereford.
You are the tail gunner, can you score enough hits on the incoming attack plane before your fuel runs out?
Runs in 16K Cassette or 32K Disk min.

CRICKET
by B. Barwick - Plymouth.
Enjoy a game of limited over cricket against the computer.
Runs in 16K Cassette or 32K Disk min.

DEFENDER
by Paul Barber - Kings Lynn.
Stop the invaders in their tracks.
Runs in 32K Cassette or Disk min.

LUNARLANDER
by Paul Barber - Kings Lynn.
Try to land on the moon without crashing, you must constantly feed the computer with thrust power details.
Runs in 16K Cassette or Disk min.

SEAWOLF
by Paul Barber - Kings Lynn.
You are Commander of a submarine and you must torpedo the enemy ships before they get you.
Runs in 32K Cassette or Disk min.

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

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 min.

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 min.

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 min.

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 min.

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 min.

FRUIT MACHINE
by Mike Nash - Radstock.
A graphic representation of a fruit machine, where you can gamble the "computer's money". Incorporates nudge and hold facilities.
Runs in 16K Cassette or 32K Disk min.

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

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.
Not XL compatible.

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

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

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

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

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

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 min.

MOONLANDER
by D. Mensing - Sutton Coldfield.
Manoeuver your craft onto the landing pads.
Runs in 16K Cassette or 32K Disk min.

LIBRARY SOFTWARE

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 min.

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

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

DOGFIGHT
by Rod Knowles - Merseyside.
A game for two players involving World War 1 bi-planes in combat.
Runs in 16K Cassette or 32K Disk min.

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

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 min.

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

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 min.

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 min.

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 min.

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

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 min.

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

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

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 min.

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

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

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 min.

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 min.

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

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

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 min.

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

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

AMERICAN CIVIL WAR
by Steve Tullett - Dalkeith.
Relive the battles of the American Civil War. Spend your money wisely, choose the correct strategy and change the course of history.
Runs in 16K Cassette or 32K Disk min.

ANAGRAM GAME
by Steve Tullett - Dalkeith.
Two player game, one inserts a word, the other has to decipher what it is once the computer has rearranged the letters.
Runs in 16K Cassette or Disk min.

ALIENOIDS
by Steve Tullett - Dalkeith.
Shoot the Alienoids with your revolving space cannon.
Runs in any size Cassette only.

BOWLING
by Jon Whitehead - Worksop.
Ten pin bowling game, uses a joystick in port 1.
16K or 48K versions cassette only.

CATS AND DOGS
by S. Walker - Leeds.
Trap the cat with your pack of dogs in this board game.
Runs in 16K Cassette or 32K Disk min.

COMPUTER WIN
by I. Mcleod - Blaydon on Tyne.
Input information on Horses, Jockeys, etc. and the computer will give the form of the race. Note this program is for fun only.
Runs in 16K Cassette orDisk min.

PENNYSLOT
by S. Fawcett - Canterbury.
Excellent slot machine game incorporating Hold and Nudge.
Runs in 32K Cassette or Disk min.

PLANETRON
by Mark & Brian Christian - Wirral.
Protect your city from the alien craft, if your first tower gun is hit you take up your reserve position in the second tower.
Runs in 32K Cassette or 48K Disk min.

SWAG
by Grahame Fairhall - Oxon.
Move your swagman around the maze, pick up the key, unlock the gold and make your way to the safe zone, avoiding the guard and his dog.
Disk or cassette versions available.
Runs in 16K Cassette or 32K Disk min.

TRAPDOOR
by R.W. Askew - Northampton.
Two player game using joysticks 1 & 2, its a race up the ladders, unlocking the trapdoors as you go, to get to the bouncing ball. Whoever gets there first can try to knock his opponent through a trapdoor with it.
Runs in 16K Cassette or 32K Disk min.

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 min.

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

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 min.

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 min.

Home Entertainment

SHAPE MATCH
by Ann Yates - Le Vesinet, France.
This program generates a selection of shapes in varying colours, and you must match the shapes. Suitable for the 2-6 age group.
Runs in 16K Cassette or 32K Disk min.

MEMORYMATCH
by Richard Chin - Llanelli.
'Simon' type game in which you have to repeat a pattern of notes and colours.
Runs in 32K Cassette or Disk min.

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

ELECTRIC SHOCK
by Steve Tullett - Dalkeith.
A 2 to 4 player game, you must get your men home without jumping on the electrified squares.
Runs in 16K Cassette or Disk min.

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 min.

SYNTHESISER
by Chris Payne - Manchester.
Program your keyboard to act as a synthesiser.
Runs in 16K Cassette or 32K Disk min.
Not XL compatible.

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

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

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 min.

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 min.

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

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

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

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

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 min.

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 min.

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

LIBRARY SOFTWARE

ATARI BINGO
by A. Irons - Selly Oak.
This program generates the bingo numbers and will check your card for winning numbers.
Runs in 16K Cassette or Disk min.

BATTLESHIPS
by Alan Wood - Walton on Thames.
Play against the computer, try to sink its navy before it sinks you!
Runs in 16K Cassette or 32K Disk min.

CAESARS CODE
by T. Hull - Leicester.
Make secret messages with this 'Secret Agents' program.
Runs in 16K Cassette or Disk min.

MICROMIND
by Jon Williamson - Bradford.
You have ten attempts to find the correct sequence of numbers.
Runs in 16K Cassette or Disk min.

OTHELLO
by Jon Williamson - Bradford.
One player game against the computer, uses joystick in port 1.
Runs in 16K Cassette or 32K Disk min.

PONTOON
by Grahame Fairall - Oxon.
Play against the computer in this well known card game.
Runs in 16K Cassette or Disk min.

Demo's

WORLD MAP
by Andrew Tullett - Dalkeith.
Map of the world, could be used as the basis for an educational program on Geography.
Runs in 16K Cassette or Disk min.

XIO DEMO
by Adrian Beesley - Gorton.
A demonstration of the Fill command on the ATARI.
Runs in 16K Cassette or Disk min.

GALLEON
by Lance Gettins - Kings Lynn.
Excellent demo of the PLOT & DRAWTO capabilities of the ATARI, depicting a Spanish Galleon on the high seas.
Runs in 16K Cassette or 32K Disk min.

EXXON
by Paul Barber - Kings Lynn.
Animated demo showing the dangers of leaking chemical pipes.
Runs in 32K Cassette or Disk min.

U.F.O.
by Paul Barber - Kings Lynn.
Graphic demo of rotating colours depicting a U.F.O.
Runs in 16K Cassette or Disk min.

EXPO
by Paul Barber - Kings Lynn.
Animated demo of a chemistry experiment.
Runs in 32K Cassette or Disk min.

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

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

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

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

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 min.

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 min.

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

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

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

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

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

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

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

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

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

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 min.

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

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

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

HILBERT AND SQUIRAL
by J. Bennett - Newcastle.
Two graphic demos in BASIC, produced using Compute! Oct. 82 PILOT EDITOR/TRANSLATOR.
Runs in 16K Cassette or Disk min.

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 min.

XMAS CARD
by A Irons - Selly Oak.
Animated Christmas scene.
Runs in 16K Cassette or 32K Disk min.

GTIA TEXT
by Frankie Smyth - Belfast.
Short article on how to add text to GTIA modes.
Runs in 16K Cassette or 32K Disk min.

MOVING DEMO
by A. Irons - Selly Oak.
Demo using colours and sounds to create movement.
Runs in 16K Cassette or 32K Disk min.

PICTURE SHOW
by Keith Berry - Birmingham.
Seven pictures created using the 'Fun with Art' cartridge from EPYX, displayed as a continuous demo.
Runs in 48K min. Disk only.

Utilities

TUTOR WRITER
by David Harry - Solihull.
This program takes information for a learning program and uses the Forced Screen Read to write the info into a program and then erases itself leaving only the tutor in memory. This can then be SAVEd.
Runs in 16K Cassette or Disk min.

LABEL PRINTER
by Peter Blackmore - Rayleigh.
Cassette based self generating data storage label routine, which has the facility to edit and retrieve, before being printed.
Runs in 16K Cassette or 32K Disk min.

DISK SPEED CHECKER
by John Attfield - Benfleet.
Check your disk drive speed is correct. This program is designed to work in the U.K.
Runs in any size Disk system.

TEXT EDITOR
by Paul Barber - Kings Lynn.
This program allows you to write & then save pages of text to disk. Print option available.
Runs in 32K Disk minimum.

TEMPERATURE CONVERSION
by Bernard West - Loughborough.
Convert Fahrenheit to Celcius and vice versa.
Runs in 16K Cassette or Disk min.

ANAGRAMS
by Keith Berry - Birmingham.
Shuffles the letters randomly in a word of any length.
Runs in 16K Cassette or Disk min.

CHARACTER GENERATOR 4
by Trevor Skeggs - Milton Keynes.
Get your custom characters into use FAST with this USR generator. No 'please waits'. No POKE's. No Strings. Up to 6 characters can be redefined from the keyboard.
Runs in 16K Cassette or Disk min.

REDEFINER
by Richard Chin - Llanelli.
Redefine up to 18 characters, the character is displayed in each of the text modes including the two multicolour text modes (ANTIC 3 & 4). A BASIC subroutine can be created containing all the logic necessary to transfer the character set and POKE in your redefined characters from DATA statements. Characters can be saved on tape or disk.
Runs in 32K Cassette or Disk min.

SUPASORT
by Peter Bryant - Maidenhead.
A multiphase sort program in BASIC with assembler inserts. Includes a PRE and POST sort phase to allow copying of files, also the sort can be normal mode or tag mode, incorporating composite keys.
Runs in 32K Cassette or Disk min.

ACCESS MINDER
by A. Lusher - Erith.
This is a small financial program which allows you to keep a check on your credit card commitments. This program is written in ATARI MICROSOFT BASIC.
Runs in 48K Disk systems only.

STOCK MARKET ANALYSER
by James Kerr - Gullane.
Keep records of stocks & shares, and use this program to analyse the best companies to get your money into!
Runs in 16K Cassette or 32K Disk min.

Q.R.A.
by Chris Barlow - Leigh.
Input QRA locations and work out distance and point value from your station.
Runs in 16K Cassette or Disk min.

ARTIST
by Martin Byfield - Birmingham.
Use your joystick to paint pictures in GR.7. Excellent utility incorporating very good player movement.
Runs in 16K Cassette or Disk min.

AUTOMATIC LINE NUMBERS
by Paul Barber - Kings Lynn.
A useful utility which automatically writes your line numbers in any size steps you wish.
Runs in 16K Cassette or Disk min.

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 min.
Not XL compatible.

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

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 min.

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 min.

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.

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 min.

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.

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 min.

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 min.

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 min.

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.

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 compatible 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.

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 min.

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

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

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

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 min.

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 min.

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 min.

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 min.

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

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

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 min.

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

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

AUTOBOOT CASSETTE COPIER
by M. Maestranzi - Finchley.
Create backup copies of your autoboot cassettes.
Any size Cassette system.

HANDWRITING
by Frankie Smyth - Belfast.
A redefined character set.
Runs in 16K Cassette or Disk min.

HEXKIT
by S. Walker - Leeds.
A useful utility for entering and manipulating HEX code.
Runs in 16K Cassette or Disk min.

MINIDOS
by Grahame Fairall - Oxon.
This program give the DOS features Directory Search, Lock, Unlock, Delete and Rename all from Basic, without deleting a program already in memory.
Runs in any size Disk only.

QUICKIE FORMAT
by Grahame Fairall - Oxon.
Use this BASIC program to format disks without the need to go to DOS.
Runs in any size Disk only.

★★★★ STAR PROGRAM ★★★★

★ USERCOMP ★
by Trevor Skeggs - Milton Keynes.
User subroutine compiler. 63 Opcodes and Directives directly incorporate your USRs into Strings and/or Data. Keystroke efficient, error detection on input (not Assembly).
Runs in 16K Cassette or 32K Disk min.
★★★★★★★★★★★★★★★★★

PRINTER DRIVER
by J. Bennett - Newcastle.
An improved printer driver for use with the Printer Interface published in ANTIC Oct. 83. Uses Ports 2, 3, & 4 and gives Graphics print feature.
Not XL compatible.
Runs in 48K Cassette or Disk min.

MINIPLAN
by J. Bennett - Newcastle.
Miniplan sheet allowing calculations on complete rows and columns, individual locations and row/column totaling. It handles numbers only but can be up to 8 characters including decimal point.
Runs in 16K Cassette or 32K Disk min.

RTTY
by Chris Barlow - Leigh on Sea.
Useful transmit and receive program for use with RTTY hardware.
Runs in 16K Cassette or Disk min.

SECTOR SAVE/LOAD
by S. Fawcett - Canterbury.
This disk utility will load and save any amount of sectors, anywhere on a disk, to or from anywhere in memory. All numbers are to be entered in HEX format.
Runs in 48K Disk systems only.
Not XL compatible.

UNIT CALCULATOR
by Steven Hind - Ravenshead.
Convert temperature, distance, volume, area or weight to/from imperial and metric.
Runs in 16K Cassette or Disk min.

VISACARD
by Keith Berry -Birmingham.
Keep a check of all your Barclay Card transactions with this useful program.
Runs in 32K min. Disk only.

80 COLUMN TEXT
by Greg Hartland - Bacons End.
Presented in the form of a demo of the possibilities of 80 character text displays. Incorporates two useful machine code routines for 80 column text and scrolling.
Runs in 32K Cassette or 48K Disk min.

Education

CHEMISTRY TUTOR
by David Harry - Solihull.
Will teach or revise the chemical symbols and valencies of the 33 commonest elements and radicals in 'O' level chemistry.
Runs in 16K Cassette or 32K Disk min.

MATHEMATICS
by Keith Berry - Birmingham.
Excellent mathematical problem setting program. Includes questions on Cubic Volume, Pythagoras Theorem, etc.
Runs in 16K Cassette or 32K Disk min.

BIBLE NAMES
by P. Brown - Newquay.
Hangman type game but designed to teach children the names of people from the Bible.
Runs in 16K Cassette or 32K Disk min.

KEYBOARD TRAINER
by H. M. Hoffman - London.
Learn to type faster with this useful program.
Runs in 16K Cassette or 32K Disk min.

NATIONAL FLAGS
by Keith Berry - Birmingham.
Flags of the world can be displayed or made into a quiz for children.
Runs in 16K Cassette or 32K Disk min.

TRACK THE ALIEN
by L. Goldsworthy - Ealing.
Keep track of the alien craft, a guessing game for children.
Runs in 16K Cassette or Disk min.

SPELLING
by L. Goldsworthy - Ealing.
Spelling game for young children.
Runs in 16K Cassette or Disk min.

CAPITALS
by Colin Marriott - Westoning.
Test your knowledge of the capitals of the world.
Runs in 16K Cassette or Disk min.

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 min.

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

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

ADDUM-UP
by Frank Silcock - Mountain Ash.
Simple sums presented in a visually entertaining way designed to hold the attention of the the very young.
Runs in 16K Cassette or 32K Disk min.

FRENCH PHRASE BOOK
by Steve & Andrew Tullett - Dalkeith.
Visiting France? Brush up on some useful phrases.
Runs in 16K Cassette or Disk min.

MONARCHS
by Keith Berry -Birmingham.
Educational program giving the kings and queens of England from 1066 to present day. Can be run in 'quiz' or 'display' mode. Includes its own redefined character set.
Runs in 32K Disk only.

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 in 16K Cassette or 32K Disk min.

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 min.

80 COLUMN TEXT DISPLAY

By Greg Hartland

The Basic program presented here consists of four main parts, two machine code routines for screen manipulation and two basic routines, one to redefine the character set and the other is a small demo to show the capabilities of the routines.

A WARNING, before anyone gets too far into the typing, if you do not have a monitor or a T.V. with a video channel, you may, due to artifacting, have difficulty reading the text produced by these routines, (altering the colours I have chosen might help, LINE 65). Don't be put off completely, however, as the scrolling routine will work on any GR.8 screen whatever it contains.

I have also shown the source listings for the 80 column and scrolling routines in case anyone would like to dig a little deeper than I will explain in this article.

Eighty Columns

The main section of coding in this routine is based largely around Keith Mayhew's excellent program 'TEXT ON GRAPHICS 8' (Issue 4), which enabled you to position standard text characters at any position on the GR.8 screen. The new program produces text in 80 columns and will support underlining actually on screen. To use the routine, you must place the text you wish to display into T$, then type POSITION X, Y where X = the horizontal position 0 to 79 and Y = the vertical position 0 to 23. Then call the routine with the command Z=USR (ADR (TEXT$)). This will place the text in T$ at the POSITION you requested on the screen.

The routine supports two types of underlining, one is switched on by the control 'U' character embedded in T$ and turned off by the control 'N' character (Underlined and Normal). The other is a forced underline which is selected by POKEing location 206 with 1 to turn it on and 0 to clear it. When the forced underline is operative EVERYTHING in T$ will be sent to the screen underlined, so it is probably good practice to clear location 206 at the very start of your program.

The actual flow of the routine is as follows: Firstly, it searches for and finds the location and length of T$, then using the POSITION statement it works out where on the screen the first character should fall. The complexity arises from the fact that 80 characters are to be plotted across the screen where there are only 40 bytes, therefore each successive character must be sent alternately to the left and then the right half of the screen byte.

Each character in the new character set has the rightmost four bits cleared, so if the character falls on an even column number 0,2 etc. the left half of the screen byte is cleared and the new character is logically 'OR'ed onto it. Why not just store the new character directly into the screen byte, I hear you asking? If there was already a character at column 3 and you wanted to add a character at column 2, then the empty right half of the new character would obliterate the character at column 3. By 'OR'ing the new character onto the screen byte you are in effect, adding the new character to what is already there. If the character falls on an odd numbered column then the character is first shifted to the right four times, this makes the left half of the character clear, and then it is 'OR'ed to the screen byte.

The routine actually writes the characters to the screen one scanline at a time, i.e. if T$ = "THE" then the routine sends the top line of the T, H, and E to the screen together, before it moves down to send scanline 2, 3 to 8. If the underline flag is set (location 206) when the routine reaches scanline 7, then it exclusive 'OR's (EOR) a solid black line to the screen, except where there is a descender, i.e. y, j, g etc. crosses the line, in which case the crossing point is cleared.

Smooth Scrolling

I have tried to keep this routine as simple to use as possible. Like the 80 column routine, it is loaded into a basic string 'SCROLL$' to be called with the statement Z=USR (ADR

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

Runs in 32K Cassette or 48K Disk

```
10 GOSUB 30000:REM INITIALISATION
12 GRAPHICS 0:POKE 752,1
17 IF PEEK(CHSET+9)=32 THEN 27
18 ? :? "INSTALLING NEW CHARACTER SET"
20 GOSUB 32500:REM 80 COLUMN CHR SET
27 IF PEEK(ADR(SCROLL$))=104 THEN 37
28 ? :? "INSTALLING SCROLLING ROUTINE"
30 GOSUB 31100:REM SCROLLING ROUTINE
37 IF PEEK(ADR(TEXT$))=104 THEN 60
38 ? :? "INSTALLING 80 COLUMN ROUTINE"
40 GOSUB 31000:REM 80 COLUMN ROUTINE
60 GRAPHICS 8+16:POKE 756,NOWTOP
65 POKE 709,2:POKE 710,8
70 GOTO 115
97 REM  SCROLLING COMMAND
98 Z=USR(ADR(SCROLL$),K,L):RETURN
99 REM  PRINTING COMMAND
100 Z=USR(ADR(TEXT$)):RETURN
110 REM  SMALL DEMO ROUTINE
115 POKE UFLAG,0
120 T$=" THIS IS A SMALL DEMO OF THE PATENT HARTLAND/MAY
HEW 80 COLUMN DISPLAY"
130 POSITION 3,0:GOSUB 100
135 K=ASC("D"):L=0:FOR J=0 TO 8:GOSUB 98:NEXT J
136 GOSUB 500
137 REM Note: In line 140 replace the two *'s with the f
ollowing:
138 REM CONTROL U before Underlined and...
139 REM CONTROL N before text.
140 T$=" IT CAN PRODUCE ON THE SCREEN,CAPITALS,lower cas
e and*Underlined*text"
150 POSITION 3,23:GOSUB 100
155 K=ASC("U"):L=11:FOR J=0 TO 11:GOSUB 98:NEXT J
156 GOSUB 500
160 K=ASC("D"):GOSUB 98:GOSUB 98
170 T$="DOWN":POSITION 35,11:GOSUB 100:GOSUB 98
180 T$="SCROLL":GOSUB 100:GOSUB 98
190 T$="CAN":GOSUB 100:GOSUB 98
200 T$="IT":GOSUB 100
205 GOSUB 500
210 K=ASC("U")
220 GOSUB 98:T$="AND":POSITION 35,17:GOSUB 100
230 GOSUB 98:T$="UP":GOSUB 100
240 GOSUB 98:T$="AGAIN":GOSUB 100
250 GOSUB 98:GOSUB 98
255 GOSUB 500
260 L=13:GOSUB 98:GOSUB 98:GOSUB 98
270 T$="FROM":POSITION 35,23:GOSUB 100:GOSUB 98
280 T$="ANY":GOSUB 100:GOSUB 98
290 T$="LINE":GOSUB 100:GOSUB 98
295 T$="POSITION":GOSUB 100
300 FOR J=0 TO 6:GOSUB 98:NEXT J
305 GOSUB 500
310 K=ASC("D"):FOR L=16 TO 13 STEP -1:FOR J=0 TO 10:GOSU
B 98:NEXT J:NEXT L
```

```
320 K=ASC("U"):L=0:FOR J=0 TO 11:GOSUB 98:NEXT J
325 GOSUB 500
330 GOTO 115
498 REM END OF DEMO
499 END
500 FOR M=0 TO 200:NEXT M:RETURN
29999 END
30000 DIM TEXT$(443),SCROLL$(304),T$(80),SPC$(80)
30010 DIM FILE$(4800),TEMP$(4800)
30020 SX=0:SY=0:FX=0:FY=1:K=ASC("U"):UFLAG=206
30100 REM
30110 NOWTOP=PEEK(106)-44
30120 RETURN
30999 REM 80 COLUMN SCREEN DISPLAY
31000 RESTORE 31010
31005 FOR J=1 TO 441:READ A:TEXT$(J,J)=CHR$(A):POSITION
32,6:? "    ";441-J:NEXT J:RETURN
31006 RETURN
31010 DATA 104,165,130,133,203,165,131,133,204,162
31011 DATA 0,160,255,200,177,203,16,251,232,192
31012 DATA 1,208,14,201,164,208,10,136,177,203
31013 DATA 200,201,84,208,2,240,20,165,203,200
31014 DATA 132,34,24,101,34,133,203,165,204,105
31015 DATA 0,133,204,169,0,240,210,202,138,162
31016 DATA 0,134,41,24,42,38,41,42,38,41
31017 DATA 42,38,41,24,101,134,133,40,165,41
31018 DATA 101,135,133,41,160,2,177,40,133,42
31019 DATA 200,177,40,133,43,200,177,40,133,32
31020 DATA 165,42,24,101,140,133,42,165,43,101
31021 DATA 141,133,43,165,84,201,24,144,1,96
31022 DATA 162,3,10,202,208,252,133,46,133,34
31023 DATA 169,0,133,47,133,35,162,5,24,38
31024 DATA 46,38,47,202,208,248,162,3,24,38
31025 DATA 34,38,35,202,208,248,165,46,24,101
31026 DATA 34,133,46,165,47,101,35,133,47,165
31027 DATA 46,24,101,88,133,46,165,47,101,89
31028 DATA 133,47,165,85,201,80,144,1,96,133
31029 DATA 34,169,0,133,36,24,102,34,38,36
31030 DATA 165,36,133,207,165,46,24,101,34,133
31031 DATA 46,165,47,105,0,133,47,169,0,133
31032 DATA 39,160,0,177,46,133,38,169,0,133
31033 DATA 33,165,206,133,205,164,33,177,42,201
31034 DATA 21,208,2,230,206,201,14,208,6,162
31035 DATA 0,134,205,134,206,41,127,201,32,176
31036 DATA 7,24,105,64,160,0,240,7,201,96
31037 DATA 176,3,56,233,32,133,44,169,0,133
31038 DATA 45,162,3,24,38,44,38,45,202,208
31039 DATA 248,208,184,165,45,24,109,244,2,133
31040 DATA 45,165,36,24,74,144,53,144,178,165
31041 DATA 38,41,240,133,38,164,39,177,44,166
31042 DATA 205,240,8,166,39,224,7,208,2,73
31043 DATA 240,162,4,74,202,208,252,5,38,164
31044 DATA 33,132,37,70,37,164,37,145,46,200
31045 DATA 177,46,133,38,169,0,133,36,240,43
31046 DATA 165,38,41,15,133,38,164,39,177,44
31047 DATA 166,205,240,10,166,39,224,7,208,4
31048 DATA 208,165,73,240,5,38,164,33,200,132
31049 DATA 37,70,37,164,37,145,46,133,38,169
31050 DATA 1,133,36,230,33,165,33,197,32,144
31051 DATA 152,165,46,24,105,40,133,46,165,47
31052 DATA 105,0,133,47,230,39,165,39,201,8
```

(SCROLL$) K, L) where K=ASC ("U") or ASC ("D"), Up or Down, and L = the line number from which you wish to start scrolling. For example, if you want the bottom half of the screen to scroll down a line (one character line or 8 scanlines) then you type in Z=USR (ADR (SCROLL$) ASC ("D"), 12). If you plan to use the scrolling routine more than once however, then you would be better off leaving the variables K and L as variables and having the USR call as a subroutine. Then you can redefine K and L as often as you need before calling the subroutine. This saves a lot of typing, see my demo program for details.

The speed of this routine at the moment is, I think, the best combination of speed and smoothness that the routine will allow. It moves the screen 2 scanlines at a time, for 4 consecutive moves. This is the standard setting. If neccessary it can move 1 scanline for 8 moves (very smooth), 4 scanlines for 2 moves (half as jerky as GR.0 scrolling), or 8 scanlines in 1 move (very fast, but as jerky as GR.0 scrolling).

New Character Set

When I first started considering an 80 column character set I wasn't too hopeful of its success. Primarily because of the small amount of room available for each character. The Graphics 8 screen has 320 pixels across its width, divide this by your 80 columns and you are left with 4 pixels for each character, take off one pixel on the left so that there is a space between each character and you are left with only 3 pixels.

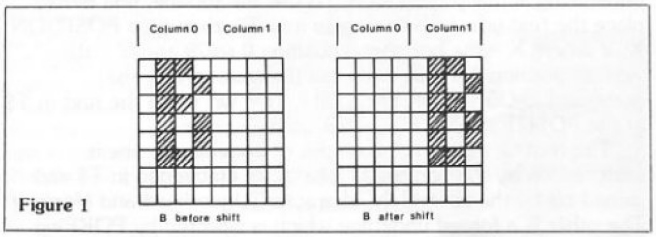


Figure 1

Sounds ridiculous doesn't it, only 3 horizontal positions to play with, to distinguish between 26 capital letters, 26 lower case letters and all numbers and punctuation marks. But when you sit down with a piece of paper and experiment a little, you find out that it is possible. See figure 1 for an example of the letter 'B' from the new character set and also the same letter after being shifted to go in an odd numbered column.

If you find any of the characters I have designed so objectionable that you must change them, they are quite easy to find and alter in the basic program. If anyone develops anything interesting from this program I would be delighted to hear about it, just write to me via the club address.

```
31053 DATA 176,8,165,207,133,36,160,1,208,196,96
31099 REM SCROLLING ROUTINE G.8
31100 RESTORE 31110
31105 FOR J=1 TO 304:READ A:SCROLL$(J,J)=CHR$(A):POSITIO
N 32,4:? "    ";304-J:NEXT J:RETURN
31106 RETURN
31110 DATA 104,104,104,133,32,104,104,201,24,144
31111 DATA 1,96,133,34,133,36,162,3,24,6
31112 DATA 34,6,36,202,208,248,169,0,133,35
31113 DATA 133,37,162,5,24,38,34,38,35,202
31114 DATA 208,248,162,3,24,38,36,38,37,202
31115 DATA 208,248,165,34,24,101,36,133,34,165
31116 DATA 35,101,37,133,35,165,34,24,101,88
31117 DATA 133,34,133,36,165,35,101,89,133,35
31118 DATA 133,37,165,88,133,38,133,40,165,89
31119 DATA 24,105,30,133,39,133,41,165,32,201
31120 DATA 85,240,5,201,68,240,99,96,165,38
31121 DATA 56,233,80,133,38,165,39,233,0,133
31122 DATA 39,169,4,133,33,165,34,24,105,80
31123 DATA 133,42,165,35,105,0,133,43,160,39
31124 DATA 177,42,145,34,136,16,249,165,34,24
```

```
31125 DATA 105,40,133,34,165,35,105,0,133,35
31126 DATA 197,39,240,4,169,0,240,213,165,34
31127 DATA 197,38,240,4,169,0,240,203,160,79
31128 DATA 169,0,145,34,136,16,251,198,33,165
31129 DATA 33,240,12,165,36,133,34,165,37,133
31130 DATA 35,169,0,240,176,96,165,34,24,105
31131 DATA 80,133,34,165,35,105,0,133,35,169
31132 DATA 4,133,33,165,38,56,233,80,133,42
31133 DATA 165,39,233,0,133,43,160,39,177,42
31134 DATA 145,38,136,16,249,165,38,56,233,40
31135 DATA 133,38,165,39,233,0,133,39,197,35
31136 DATA 240,4,169,0,240,213,165,38,197,34
31137 DATA 240,4,169,0,240,203,160,119,169,0
31138 DATA 145,36,136,16,251,198,33,165,33,240
31139 DATA 12,165,40,133,38,165,41,133,39,169
31140 DATA 0,240,176,96
31141 REM
31142 REM
31143 REM
32499 REM  80 COLUMN CHARACTER SET
32500 CHSET=NOWTOP*256
32501 RESTORE 32505
32502 READ A:IF A=-1 THEN POSITION 32,2:? "    0":RE
TURN
32503 FOR J=0 TO 7:READ B:POKE CHSET+A*8+J,B:POSITION 32
,2:? "    ";122-A:NEXT J
32504 GOTO 32502
32505 DATA 1,0,32,32,32,0,32,0,0
32506 DATA 2,0,80,80,0,0,0,0,0
32507 DATA 3,0,80,240,80,240,80,0,0
32508 DATA 4,32,48,64,32,16,96,32,0
32509 DATA 5,0,64,16,32,64,16,0,0
32510 DATA 6,0,32,32,112,32,32,0,0
32511 DATA 7,0,32,32,0,0,0,0,0
32512 DATA 8,0,48,96,64,96,48,0,0
32513 DATA 9,0,96,48,16,48,96,0,0
32514 DATA 10,0,80,112,32,112,80,0,0
32515 DATA 11,0,32,32,112,32,32,0,0
32516 DATA 12,0,0,0,0,32,32,64,0
32517 DATA 13,0,0,0,112,0,0,0,0
32518 DATA 14,0,0,0,0,96,96,0,0
32519 DATA 15,0,16,16,32,64,64,0,0
32520 DATA 16,0,112,80,80,80,112,0,0
32521 DATA 17,0,32,32,32,32,32,0,0
32522 DATA 18,0,112,16,112,64,112,0,0
32523 DATA 19,0,112,16,112,16,112,0,0
32524 DATA 20,0,64,80,112,16,16,0,0
32525 DATA 21,0,112,64,112,16,112,0,0
32526 DATA 22,0,112,64,112,80,112,0,0
32527 DATA 23,0,112,16,32,64,64,0,0
32528 DATA 24,0,112,80,112,80,112,0,0
32529 DATA 25,0,112,80,112,16,112,0,0
32530 DATA 26,0,32,32,0,32,32,0,0
32531 DATA 27,0,32,32,0,32,32,64,0
32532 DATA 28,0,16,32,64,32,16,0,0
32533 DATA 29,0,0,112,0,112,0,0,0
32534 DATA 30,0,64,32,16,32,64,0,0
32535 DATA 31,0,96,16,32,0,32,0,0
32536 DATA 32,0,96,144,176,80,48,0,0
32537 DATA 33,0,32,80,80,112,80,0,0
32538 DATA 34,0,96,80,96,80,96,0,0
32539 DATA 35,0,32,80,64,80,32,0,0
32540 DATA 36,0,96,80,80,80,96,0,0
32541 DATA 37,0,112,64,112,64,112,0,0
32542 DATA 38,0,112,64,96,64,64,0,0
32543 DATA 39,0,48,64,80,80,48,0,0
32544 DATA 40,0,80,80,112,80,80,0,0
32545 DATA 41,0,32,32,32,32,32,0,0
32546 DATA 42,0,16,16,16,80,32,0,0
32547 DATA 43,0,80,80,96,80,80,0,0
32548 DATA 44,0,64,64,64,64,112,0,0
32549 DATA 45,0,80,112,112,80,80,0,0
32550 DATA 46,0,80,112,112,112,80,0,0
32551 DATA 47,0,112,80,80,80,112,0,0
32552 DATA 48,0,96,80,96,64,64,0,0
32553 DATA 49,0,112,80,80,80,96,16,0
32554 DATA 50,0,96,80,96,80,80,0,0
32555 DATA 51,0,48,64,32,16,96,0,0
32556 DATA 52,0,112,32,32,32,32,0,0
32557 DATA 53,0,80,80,80,80,112,0,0
32558 DATA 54,0,80,80,80,32,32,0,0
32559 DATA 55,0,80,80,112,112,80,0,0
32560 DATA 56,0,80,80,32,80,80,0,0
32561 DATA 57,0,80,80,32,32,32,0,0
32562 DATA 58,0,112,16,32,64,112,0,0
32563 DATA 59,0,112,96,96,96,112,0,0
32564 DATA 60,0,64,64,32,16,16,0,0
32565 DATA 61,0,112,48,48,48,112,0,0
32566 DATA 62,0,0,32,32,80,80,0,0
32567 DATA 63,0,0,0,0,0,0,0,240
32568 DATA 97,0,0,0,48,80,48,0,0
32569 DATA 98,0,64,64,96,80,96,0,0
32570 DATA 99,0,0,0,48,64,48,0,0
32571 DATA 100,0,16,16,48,80,48,0,0
32572 DATA 101,0,0,32,80,96,48,0,0
32573 DATA 102,0,32,64,96,64,64,0,0
32574 DATA 103,0,0,0,48,80,48,16,32
32575 DATA 104,0,64,64,96,80,80,0,0
32576 DATA 105,0,0,32,0,32,32,0,0
32577 DATA 106,0,0,0,16,16,16,16,32
32578 DATA 107,0,64,64,80,96,80,0,0
32579 DATA 108,0,64,64,64,64,96,0,0
32580 DATA 109,0,0,0,112,112,80,0,0
32581 DATA 110,0,0,0,112,80,80,0,0
32582 DATA 111,0,0,0,112,80,112,0,0
32583 DATA 112,0,0,0,112,80,112,64,64
32584 DATA 113,0,0,0,112,80,112,16,16
32585 DATA 114,0,0,0,112,64,64,0,0
32586 DATA 115,0,0,0,48,80,176,0,0
32587 DATA 116,0,32,32,48,32,48,0,0
32588 DATA 117,0,0,0,80,80,112,0,0
32589 DATA 118,0,0,0,80,80,32,0,0
32590 DATA 119,0,0,0,80,112,112,0,0
32591 DATA 120,0,0,0,80,32,80,0,0
32592 DATA 121,0,0,0,80,80,112,16,48
32593 DATA 122,0,0,0,112,32,112,0,0
32594 DATA 66,0,0,0,0,0,0,0,0
32595 DATA 78,0,0,0,0,0,0,0,0
32596 DATA 85,0,0,0,0,0,0,0,0
32597 DATA -1
```

```
10 ;Graphics 8 character plotter f
or use with eighty column screen
20       .OPT NO EJECT ;No page nu
mbers
30 ;
40 ;
50 ;O.S. Equates.
60 ;
70 ROWCRS =  $54
80 COLCRS =  $55
90 SAVMSC =  $58
0100 CHBAS = $02F4
0110 ;
0120 ;BASIC 'strings' equates.
0130 ;
0140 VNTP =  $82
0150 VVTP =  $86
0160 STARP = $8C
0170 ;
0180 ;Program variables equates.
0190 ;
0200 STRLEN = $20
0210 CHRCNT = $21
0220 TEMP =  $22
0230 OFFSET = $24
0240 BYTE1 = $25
0250 BYTE2 = $26
0260 LINCNT = $27
0270 DATBLK = $28
0280 STRLOC = $2A
0290 CHAR =  $2C
0300 SCREEN = $2E
0310 CHR =   $CB
0320 UFLAG = $CD
0330 TFLAG = $CE     ;206 Decmset
from basic to force underline,clea
r for triggered underline
0340 TMPOFF = $CF
0350      *=  $4000
0360      PLA
0370 ;
0380 ;Move variable name table poi
nter (VNTP) to our page zero point
er (CHR).
0390 ;
0400      LDA VNTP
0410      STA CHR
0420      LDA VNTP+1
0430      STA CHR+1
0440 ;'X' is used for string count
.
0450      LDX #$00
0460 NXNAM LDY #$FF
0470 NXCHR INY
0480      LDA (CHR),Y
0490      BPL NXCHR
0500      INX
0510      CPY #$01
0520      BNE INCNAM
0530      CMP #$A4    ;ASCII "$"(24
) WITH HIGH BIT SET ( +$80)
0540      BNE INCNAM
0550      DEY
0560      LDA (CHR),Y
0570      INY
0580      CMP #'T
0590      BNE INCNAM
0600      BEQ FNDNAM
0610 ;Points to next variable name
.
0620 INCNAM LDA CHR
0630      INY
0640      STY TEMP
0650      CLC
0660      ADC TEMP
0670      STA CHR
0680      LDA CHR+1
0690      ADC #$00
0700      STA CHR+1
0710      LDA #$00
0720      BEQ NXNAM
0730 ;Found position in name table
.
0740 ;
0750 ;Now find string length and s
tarting address.
0760 ;
0770 FNDNAM DEX
0780      TXA
0790      LDX #$00
0800      STX DATBLK+1
0810      CLC
0820      ROL A
0830      ROL DATBLK+1
0840      ROL A
0850      ROL DATBLK+1
0860      ROL A
0870      ROL DATBLK+1
0880      CLC
0890      ADC VVTP
0900      STA DATBLK
0910      LDA DATBLK+1
0920      ADC VVTP+1
0930      STA DATBLK+1
0940      LDY #$02
0950      LDA (DATBLK),Y
0960      STA STRLOC
0970      INY
0980      LDA (DATBLK),Y
0990      STA STRLOC+1
1000      INY
1010      LDA (DATBLK),Y
1020      STA STRLEN
1030      LDA STRLOC
1040      CLC
1050      ADC STARP
1060      STA STRLOC
1070      LDA STRLOC+1
1080      ADC STARP+1
1090      STA STRLOC+1
1100 ;Found string and length.
1110 ;
1120 ;Now add column and row to sc
reen start address.
1130      LDA ROWCRS
1140      CMP #24
1150      BCC CNT
1160      RTS          ;RETURN TO BA
SIC
1170 CNT LDX #8
1180 MUL0 ASL A
1190      DEX
1200      BNE MUL0
1210      STA SCREEN
1220      STA TEMP
1230      LDA #$00
1240      STA SCREEN+1
1250      STA TEMP+1
1260      LDX #$05
1270 MUL1 CLC
1280      ROL SCREEN
1290      ROL SCREEN+1
1300      DEX
1310      BNE MUL1
1320      LDX #$03
1330 MUL2 CLC
1340      ROL TEMP
1350      ROL TEMP+1
1360      DEX
1370      BNE MUL2
1380      LDA SCREEN
1390      CLC
1400      ADC TEMP
1410      STA SCREEN
1420      LDA SCREEN+1
1430      ADC TEMP+1
1440      STA SCREEN+1
1450      LDA SCREEN
1460      CLC
1470      ADC SAVMSC
1480      STA SCREEN
1490      LDA SCREEN+1
1500      ADC SAVMSC+1
1510      STA SCREEN+1
1520      LDA COLCRS
1530      CMP #80
1540      BCC CNT2
1550      RTS          ;RETURN TO BA
SIC
1560 CNT2 STA TEMP
1570      LDA #$00
1580      STA OFFSET
1590      CLC
1600      ROL TEMP     ;DIVIDE COLUM
N BY TWO
1610      ROL OFFSET  ;ANE IF ODD S
ET OFFSETTO 1
1620      LDA OFFSET
1630      STA TMPOFF
1640      LDA SCREEN
1650      CLC
1660      ADC TEMP
1670      STA SCREEN
```

```
1680      LDA SCREEN+1
1690      ADC #$00
1700      STA SCREEN+1
1710 ;Reset scan line counter.
1720      LDA #$00
1730      STA LINCNT
1740 ;Main Program loop.
1750 OTLOOP LDY #$00
1760      LDA (SCREEN),Y
1770      STA BYTE2
1780 ;Reset Character Counter
1790      LDA #$00
1800      STA CHRCNT
1810 ;Scanline Plotting Loop
1820 INLOOP LDA TFLAG ;If last cha
racter was ctrl-u
1830      STA UFLAG   ;then set UFL
AG
1840      LDY CHRCNT
1850      LDA (STRLOC),Y ;Look for
these characters
1860      CMP #'      ';CTRL-U
1870      BNE NORM0
1880      INC TFLAG   ;Set underlin
e flag next time
1890 NORM0 CMP #'     ';CTRL-N
1900      BNE NORM1
1910      LDX #$00
1920      STX UFLAG   ;Clear both f
lags
1930      STX TFLAG
1940 NORM1 AND #$7F
1950      CMP #32
1960      BCS N1
1970      CLC
1980      ADC #64
1990      LDY #$00
2000      BEQ N2
2010 N1   CMP #96
2020      BCS N2
2030      SEC
2040      SBC #32
2050 N2   STA CHAR
2060      LDA #$00
2070      STA CHAR+1
2080      LDX #$03
2090 MUL3 CLC
2100      ROL CHAR
2110      ROL CHAR+1
2120      DEX
2130      BNE MUL3
2140 SKIP2 BNE OTLOOP ;2nd part of
 indirect jump back to start
2150      LDA CHAR+1
2160      CLC
2170      ADC CHBAS
2180      STA CHAR+1
2190      LDA OFFSET
2200      CLC
2210      LSR A
2220      BCC LOOPEV  ;Even Column
Routine
2230 SKIPLP BCC INLOOP
2240 LOOPOD LDA BYTE2
2250      AND #$F0
2260      STA BYTE2
2270      LDY LINCNT
2280      LDA (CHAR),Y
2290      LDX UFLAG   ;Check underl
ine flag
2300      BEQ NORM2
2310      LDX LINCNT  ;If on 7th sc
anline
2320      CPX #7
2330      BNE NORM2
2340      EOR #$F0    ;Then underli
ne
2350 NORM2 LDX #$04
2360 SHF1 LSR A
2370      DEX
2380      BNE SHF1
2390      ORA BYTE2
2400      LDY CHRCNT
2410      STY BYTE1
2420      LSR BYTE1
2430      LDY BYTE1
2440      STA (SCREEN),Y
2450      INY
2460      LDA (SCREEN),Y
2470      STA BYTE2   ;Ready for ne
xt character
2480      LDA #$00
2490      STA OFFSET
2500      BEQ FINAL
2510 LOOPEV LDA BYTE2
2520      AND #$0F
2530      STA BYTE2
2540      LDY LINCNT
2550      LDA (CHAR),Y
2560      LDY UFLAG   ;Check underl
ine flag
2570      BEQ NORM3
2580      LDX LINCNT  ;If on 7th sc
anline
2590      CPX #7
2600      BNE NORM3
2610 SKIP BNE SKIP2
2620      EOR #$F0    ;Then underli
ne
2630 NORM3 ORA BYTE2
2640      LDY CHRCNT
2650      INY
2660      STY BYTE1
2670      LSR BYTE1
2680      LDY BYTE1
2690      STA (SCREEN),Y
2700      STA BYTE2   ;Ready for ne
xt character
2710      LDA #$01
2720      STA OFFSET
2730 FINAL INC CHRCNT
2740      LDA CHRCNT
2750      CMP STRLEN
2760      BCC SKIPLP
2770      LDA SCREEN
2780      CLC
2790      ADC #40
2800      STA SCREEN
2810      LDA SCREEN+1
2820      ADC #$00
2830      STA SCREEN+1
2840      INC LINCNT
2850      LDA LINCNT
2860      CMP #$08
2870      BCS RETURN
2880      LDA TMPOFF
2890      STA OFFSET
2900      LDY #$01
2910      BNE SKIP    ;1st part of
indirect jump back to start
2920 RETURN RTS       ;Return to BA
SIC
```

```
10 ; Graphics 8 scrolling routine
for use with BASIC
20 ; Call using Z=USR(ADR(SCROLL$)
,"U",#N); Where N= lineno,U=UP,D=D
OWN
30 ;
40 ; By Greg Hartland
50 ;
60 ; Equates
70 ;
80 SAVMSC =  $58
90 DIR    =  $20
0100 COUNTER = $21
0110 TSCREEN = $22
0120 TEMPU = $24
0130 BSCREEN = $26
0140 TEMPD = $28
0150 MEMLOC = $2A
0160 ;
0170      *=  $4000
0180      PLA
0190      PLA
0200      PLA
0210      STA DIR
0220      PLA
0230      PLA
0240      CMP #24
0250      BCC CNT
0260      RTS
0270 CNT STA TSCREEN
0280      STA TEMPU
0290      LDX #$03
0300 MUL0 CLC
0310      ASL TSCREEN
0320      ASL TEMPU
0330      DEX
0340      BNE MUL0
0350      LDA #$00
0360      STA TSCREEN+1
0370      STA TEMPU+1
```

```
0380     LDX #$05
0390 MUL1 CLC
0400     ROL TSCREEN
0410     ROL TSCREEN+1
0420     DEX
0430     BNE MUL1
0440     LDX #$03
0450 MUL2 CLC
0460     ROL TEMPU
0470     ROL TEMPU+1
0480     DEX
0490     BNE MUL2
0500     LDA TSCREEN
0510     CLC
0520     ADC TEMPU
0530     STA TSCREEN
0540     LDA TSCREEN+1
0550     ADC TEMPU+1
0560     STA TSCREEN+1
0570     LDA TSCREEN
0580     CLC
0590     ADC SAVMSC
0600     STA TSCREEN
0610     STA TEMPU
0620     LDA TSCREEN+1
0630     ADC SAVMSC+1
0640     STA TSCREEN+1
0650     STA TEMPU+1
0660     LDA SAVMSC
0670     STA BSCREEN
0680     STA TEMPD
0690     LDA SAVMSC+1
0700     CLC
0710     ADC #30     ;7680 (30 * 2
56) Bytes
0720     STA BSCREEN+1
0730     STA TEMPD+1
0740 ;
0750     LDA DIR
0760     CMP #'U     '
0770     BEQ SCRLPU
0780     CMP #'D     '
0790     BEQ SCRLPD
0800     RTS
0810 SCRLPU LDA BSCREEN
0820     SEC
0830     SBC #80
0840     STA BSCREEN
0850     LDA BSCREEN+1
0860     SBC #$00
0870     STA BSCREEN+1
0880     LDA #$04
0890     STA COUNTER
0900 SCRLP0 LDA TSCREEN
0910     CLC
0920     ADC #80
0930     STA MEMLOC
0940     LDA TSCREEN+1
0950     ADC #$00
0960     STA MEMLOC+1
0970     LDY #39
0980 SCRLP1 LDA (MEMLOC),Y
0990     STA (TSCREEN),Y
1000     DEY
1010     BPL SCRLP1
1020     LDA TSCREEN
1030     CLC
1040     ADC #40
1050     STA TSCREEN
1060     LDA TSCREEN+1
1070     ADC #$00
1080     STA TSCREEN+1
1090     CMP BSCREEN+1
1100     BEQ CHECK2
1110     LDA #$00
1120     BEQ SCRLP0
1130 CHECK2 LDA TSCREEN
1140     CMP BSCREEN
1150     BEQ LSTRWU
1160     LDA #$00
1170     BEQ SCRLP0
1180 LSTRWU LDY #79
1190     LDA #$00
1200 SCRLP2 STA (TSCREEN),Y
1210     DEY
1220     BPL SCRLP2
1230     DEC COUNTER
1240     LDA COUNTER
1250     BEQ RET1
1260     LDA TEMPU
1270     STA TSCREEN
1280     LDA TEMPU+1
1290     STA TSCREEN+1
1300     LDA #$00
1310     BEQ SCRLP0
1320 RET1 RTS
1330 SCRLPD LDA TSCREEN
1340     CLC
1350     ADC #80
1360     STA TSCREEN
1370     LDA TSCREEN+1
1380     ADC #$00
1390     STA TSCREEN+1
1400     LDA #$04
1410     STA COUNTER
1420 SCRLP3 LDA BSCREEN
1430     SEC
1440     SBC #80
1450     STA MEMLOC
1460     LDA BSCREEN+1
1470     SBC #$00
1480     STA MEMLOC+1
1490     LDY #39
1500 SCRLP4 LDA (MEMLOC),Y
1510     STA (BSCREEN),Y
1520     DEY
1530     BPL SCRLP4
1540     LDA BSCREEN
1550     SEC
1560     SBC #40
1570     STA BSCREEN
1580     LDA BSCREEN+1
1590     SBC #$00
1600     STA BSCREEN+1
1610     CMP TSCREEN+1
1620     BEQ CHECK3
1630     LDA #$00
1640     BEQ SCRLP3
1650 CHECK3 LDA BSCREEN
1660     CMP TSCREEN
1670     BEQ LSTRWD
1680     LDA #$00
1690     BEQ SCRLP3
1700 LSTRWD LDY #119
1710     LDA #$00
1720 SCRLP5 STA (TEMPU),Y
1730     DEY
1740     BPL SCRLP5
1750     DEC COUNTER
1760     LDA COUNTER
1770     BEQ RET2
1780     LDA TEMPD
1790     STA BSCREEN
1800     LDA TEMPD+1
1810     STA BSCREEN+1
1820     LDA #$00
1830     BEQ SCRLP3
1840 RET2 RTS
```

CONTACT

SCOTLAND
Hillfoots Computer Club based in ALVA (Clacks). New members welcome contact Mr. J. Crossan, 39 Menteith Court, Alloa, for details.

SCOTLAND
John Thomson, 23 Waverley Crescent, Riverside, Stirling.

STAFFS
Has anyone been having trouble with the DIG-DUG cartridge! Does anyone know if there is a bug with this cartridge? Contact A. Clarke, 7 Carina Gardens, Smallthorne, Stoke-on-Trent.

SOUTH YORKS
As a Registered Disabled Person I have many hours available to type in those great listings in the club newsletter, I do a few for my friends already and would be willing to do the same for other members (a minute fee is accepted). I would also like to hear from others on ATARI matters in general, why not drop me a line! James Stevens, 29 Chancel Row, Sheffield, S2 5LL.

KENT
CANTARI. For information on the Canterbury and District Atari Computer Club contact Stephen Fawcett, 6 Wife of Bath Hill, Canterbury, Kent.

MERSEYSIDE
I want to contact Adventurers with a view to swapping hints and tips, write to me at 24 Oakdene Road, Anfield, Liverpool, L4 2SR, my name is Mike Lynch.

LIGHT PEN

by Chris Barlow and Martin Taylor

Being Atari owners we all possess a computer with the ability to accept a light pen input. But if you have ever tried to obtain a ready built unit, you may have been amazed at the cost and lack of available sources. This is partly due to the lack of software that the use of this device requires, and the difficulty in manufacturing a reliable piece of hardware. Some manufacturers have attempted to produce such a device but, due to marketing considerations i.e. cost and predicted sales, the resulting hardware leaves a lot to be desired. In this article we present a Light Pen which should cost, in components, less than half the price of a commercially available unit with, in our opinion, a superior performance.

Method

To explain how a Light Pen works, you must first have an idea of how the television picture system is generated. A TV picture is basically constructed from a number of lines produced on the phosphor coating on the inside of the screen. The original TV system used 405 lines, but today 625 lines is the accepted UK standard. However, the Atari does not use all 625 lines. The phosphor on the screen will glow when electrons, produced from the electron gun, strike it. This will produce a single spot of light on the TV screen. To produce lines it is necessary to deflect the electron beam across from left to right, thus obtaining the horizontal or X axis. When the line has reached the right-hand side of the screen it is then deflected back to the left-hand side of the picture and down slightly to produce the next scan starting position. During the return period of scan, the electron gun output is blanked in order not to generate spurious lines. The downward scanning, or Y axis, continues until all 625 lines have been drawn, at which time the beam is made to return back to the top starting position. This is an over simplified description and, in reality, the total procedure is much more involved.

The Light Pen is designed so it can detect a pulse of light coming from the screen. The computer has the job of determining the X and Y coordinates of this light pulse. These values are obtained from the internal register set of the Antic Display processor. Since the position of the light pulse on the screen is directly related to the time it took to get there from the beginning of the first scan position, the hardware can determine X and Y values and store these in two hardware registers. When programming in Basic the X and Y values are obtained by PEEKing locations 564 for X and 565 for Y. The user's software has then to interpret

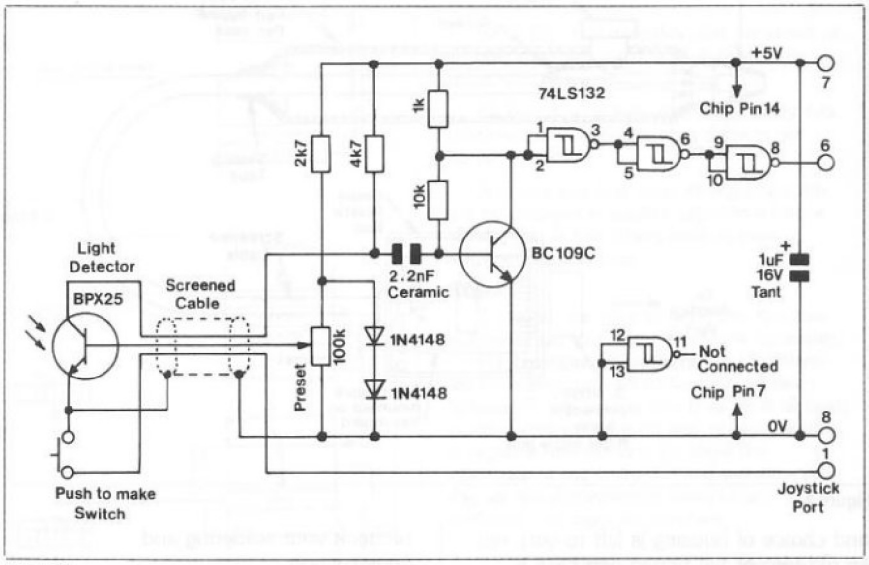


Figure 1

these values in order to obtain screen position related values. The horizontal, or X, location (564) will return a value of 78 for the extreme left-hand side of the screen, increasing in increments of one, up to a value of 227. Then something rather strange happens, the value jumps to zero and then increments up to a final value of 8 for the extreme right-hand side. This is a fault due to the Atari hardware and not to the Light Pen. It appears that Atari are not concerned with this fault since it has not been corrected in their new range of computers. Although a problem, it can be rectified by software means. The vertical, or Y, location (565) will return a value of 16 for the extreme top of the display, incrementing by one to value of 111 at the extreme bottom of the display. This register appears to function correctly. The values stored at these two locations are updated when any of the four joystick trigger inputs are used.

Circuit

To attempt to construct this project, some knowledge of electronics and the ability to solder will be necessary. As can be seen, in the circuit diagram (figure 1), there are very few components necessary to obtain a working Light Pen. The most important is the Light Detector. It must have good sensitivity and fast reaction characteristics. The BPX25 photo-transistor meets both requirements, at a modest cost. This device is equipped with its own built-in optical lens, which is made of glass. This point is worth noting, since, if direct contact is made with the glass of the TV screen, scoring may occur. To prevent this the BPX25 should be recessed into a plastic tube of some description. To obtain maximum sensitivity and operating speed, it is necessary to bias the base of the transistor. The voltage required is quite small, approximately 0.5 Volt. This voltage is adjustable by the 100k preset from 1.2 Volts down to 0 Volts. In practice the preset wiper position comes out about half way round its travel. The 1.2 Volts at the top end of the preset is generated by two silicon diodes in series, and forward biased. The current through the diodes is limited by the 2.7k resistor connected to the +5 Volt supply, taken from pin 7 of the joystick port.

When the photo-transistor detects a light pulse, the amount of current flowing through it changes. The current is limited through the device by the 4.7k resistor in its collector circuit. These changes in current cause a voltage change at the collector of the photo-transistor. The voltage pulses are then coupled, via a 2.2nF ceramic capacitor, into the base circuit of the BC109C transistor. This device performs the necessary level change to obtain TTL logic levels. The final stage of shaping the pulse is achieved by using a 74LS132, a quad two input Nand Schmitt Trigger. As can be seen, only three of the four gates are used. The final component in the circuit is a 1uF 16 Volt tantalum bead capacitor across the supply rails, which removes any spurious noise on the power lines. The output of the final gate is fed to pin 6 of the joystick port. The ground connection is made to pin 8.

In the prototype, a push-to-make switch was used as a trigger for the Light Pen. The switch was simply connected between pin 1 and pin 8 of the joystick port. The final construction

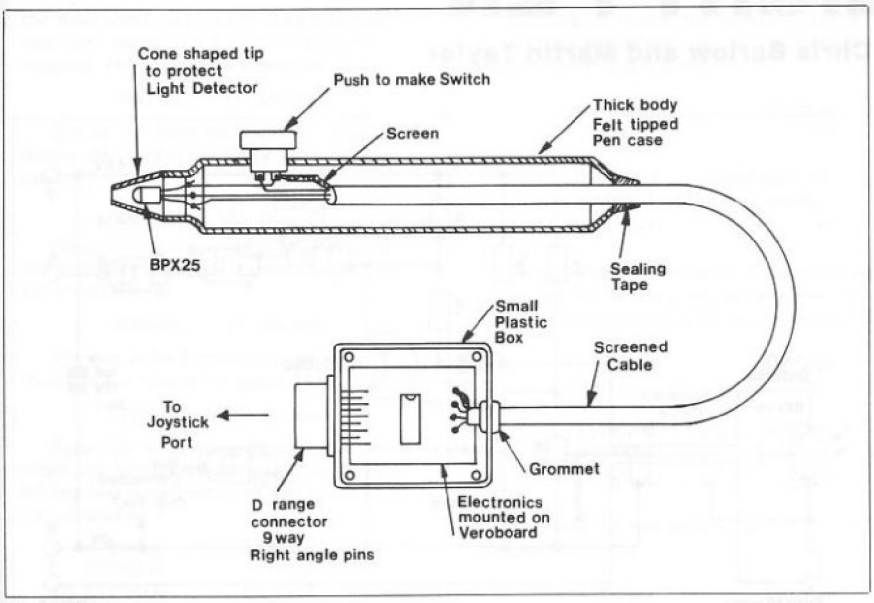


Figure 2

and choice of housing is left to you, but an old biro or felt-tipped pen case is ideal for the pen itself, and the electronics can be housed in a small plastic box (see figure 2). An electronic consideration though is that the cable linking the photo-transistor to the circuit board must be screened to prevent stray interference pickup. The prototype used 4 core overall screened cable, but individual screened audio cable is quite acceptable. The electronics can be constructed on a small piece of Vero board, or for the more ambitious you could etch your own PCB. Connection to the joystick is via a standard D-Type 9-way connector.

Programs

Included in this article are three very simple programs, the first of which is used to set up the 100k preset in the circuit. In all three programs we have used joystick port 1, this is because of the Light Pen's switch controlling the value of STICK (0). However, the Light Pen will work in any of the four joystick ports. Program 1 is a simple drawing utility which will produce lines or dots depending upon the state of the function keys. Holding Select down will put the drawing program into dot mode, and the Option key will clear the screen and reset the starting position to the current pen position. Pressing the Light Pen's own button will produce continuous line drawing. To set a new starting point simply place the pen on the screen and press the Select key. To adjust the preset to obtain the correct results simply hold the Light Pen against the screen, press the switch on the Light Pen and move the pen slowly. If the line does not trace the movement, adjust the preset until it does. If you cannot obtain a satisfactory result, try increasing the brightness and contrast controls on your TV. If there is still no response recheck your soldering and construction.

Program 2 is an example of how a Light Pen can be used for Menu-Driven software. Position the pen over the number you wish to choose and press the Light Pen switch. If all is well, a tone will be heard and your selection will be shown at the bottom of the screen.

The final program (Program 3) is a very simple musical instrument, in which you can select both volume and pitch. The sound will only be present whilst pressing the Light Pen switch. The display on the screen is a matrix of square dots with volume increasing down the screen and pitch increasing across the screen, right to left.

In conclusion, we must point out, that the programs shown are by no means good examples of what can be achieved, but are adequate for testing purposes and demonstrating the principles behind Light Pen Software implementation. When writing your own software, you must bear in mind where the screen is dark, no information can be detected by the Light Pen. It is hoped that in the future we will see more software using Light Pens, so get writing!

Program 1

```
10 GRAPHICS 24:COLOR 1
20 X=PEEK(564):X=X-155+X:IF X<1 THE
N X=1
30 Y=PEEK(565):Y=Y-30+Y:IF Y>190 TH
EN Y=190
40 IF PEEK(53279)=3 THEN GOSUB 80
50 IF PEEK(53279)=5 THEN PLOT X,Y:G
OTO 60
60 IF STICK(0)<>15 THEN DRAWTO X,Y
70 GOTO 20
80 GRAPHICS 24:COLOR 1
90 PLOT X,Y:RETURN
```

Program 2

```
10 REM MENU
20 GRAPHICS 2+16:SETCOLOR 0,0,12:SE
TCOLOR 4,4,1
30 PRINT #6;"___ atari 1"
40 PRINT #6;"___ atari 2"
50 PRINT #6;"___ ATARI 3"
60 PRINT #6;"___ ATARI 4"
70 PRINT #6;"___ atari 5"
80 PRINT #6;"___ atari 6"
90 PRINT #6;"___ ATARI 7"
100 PRINT #6;"___ ATARI 8"
110 PRINT #6;"___ atari 9"
120 PRINT #6;"___ atari 10"
130 IF STICK(0)<>15 THEN 150
140 GOTO 130
150 LET I=PEEK(565)
160 IF I<18 OR I>94 THEN 150
170 IF I=18 OR I=19 OR I=20 OR I=21
 OR I=22 THEN M=1:GOSUB 280
180 IF I=26 OR I=27 OR I=28 OR I=29
 OR I=30 THEN M=2:GOSUB 280
190 IF I=34 OR I=35 OR I=36 OR I=37
 OR I=38 THEN M=3:GOSUB 280
200 IF I=42 OR I=43 OR I=44 OR I=45
 OR I=46 THEN M=4:GOSUB 280
210 IF I=50 OR I=51 OR I=52 OR I=53
 OR I=54 THEN M=5:GOSUB 280
220 IF I=57 OR I=58 OR I=59 OR I=60
 OR I=61 THEN M=6:GOSUB 280
230 IF I=65 OR I=66 OR I=67 OR I=68
 OR I=69 THEN M=7:GOSUB 280
240 IF I=74 OR I=75 OR I=76 OR I=77
 OR I=78 THEN M=8:GOSUB 280
250 IF I=82 OR I=83 OR I=84 OR I=85
 OR I=86 THEN M=9:GOSUB 280
260 IF I=90 OR I=91 OR I=92 OR I=93
 OR I=94 THEN M=10:GOSUB 280
270 GOTO 130
280 IF MM=M THEN RETURN
290 POSITION 4,11:PRINT #6;"ATARI="
;M;" "
300 FOR V=15 TO 0 STEP -1:SOUND 0,M
*10,10,V:NEXT V:LET MM=M:RETURN
```

Program 3

```
10 GRAPHICS 4+16:COLOR 1
20 SETCOLOR 4,2,3:SETCOLOR 0,0,15
30 FOR Y=0 TO 47 STEP 4
40 FOR X=0 TO 70 STEP 4
50 PLOT X,Y
60 NEXT X:NEXT Y
70 IF STICK(0)<>15 THEN 90
80 SOUND 0,0,0,0:GOTO 70
90 SOUND 0,PEEK(564)/3,10,PEEK(565)
/10
100 GOTO 70
```

REVIEWS

JAVA JIM
A review by D.A. Dodson.

Shortly to be released, June the 11th to be precise, is "JAVA JIM", written by Kevin Buckner and produced by 'Creative Sparks' of Thorn EMI fame, and in case you have not figured it out already it's all about a little chap called Jim on the island of Java.

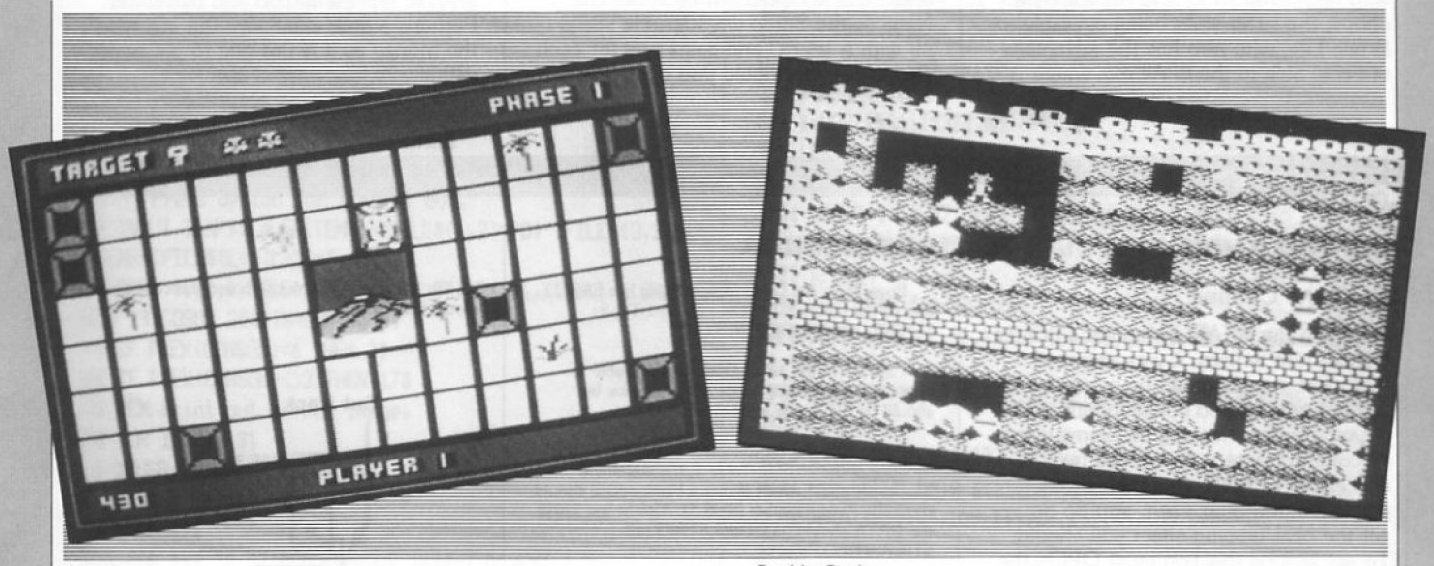
The object of the game is to move Jim around a grid style screen turning squares into holes in the search for treasure. This is achieved by moving Jim onto a square and if left for a second, either an empty hole is revealed or an item of treasure. Moving rapidly over a square leaves it untouched. "Sounds pretty straightforward" I hear you say, not so!

Casting one's mind back into the annals of history, you may recall a volcano by the name of Krakatoa. Well, sitting in the middle of the screen is a Volcano which regularly spits out lumps of lava, which not only fill in your freshly dug holes, but seem to have some sort of fiendish homing system, because the lumps always seem to land a "stone's throw," sorry, "lava's throw" away and unless you keep moving its a fair bet that you'll get clobbered. The result of such an encounter is the loss of one of your five lives. Also, whilst dodging and digging, you will encounter "Speed Spiders" and "Slug Bugs". Speed Spiders will paralyse you if touched, which in turn increases your chance of being hit by the lumps of lava. The only way to be rid of the Speed Spider is to catch it when it's green, and this also applies to the Slug Bug, which meanders around the screen planting grass and palm trees which have to be dug up prior to digging for treasure. The Slug Bug however, is deadly if touched when Red.

Clearing a screen by collecting the treasure, in the order displayed at the top of the screen, transforms the Volcano into a flight of stairs to the next level of play. The level of play however, is not sequential and is selected by hitting the fire button when the level number is flashed on the screen. A mistake and you are on a level other than intended. On the higher levels there are a limited number of digs, and once used up, you must manoeuvre Jim over a "Warp Hole", otherwise used for jumping large areas of screen. This will result in Jim being sent to a cave to chase a shovel, each touch of the shovel giving back digging power, meanwhile being pursued by snakes and fungi.

There are nine levels of play and I can only clear the first three, so there is plenty of challenge, not to mention action. The graphics are very tidy and the game has the facility for one or two players. The only things I did not like about the game, were having to restart the current level if I lost a Java Jim, and having to keep moving to avoid being hit by various hazards which, all things considered, are minor complaints.

Java Jim will be sold on cassette and has comprehensive instructions, and with an attractive price tag of £8.95 I found the game to be fast moving and addictive. Let's hope more software of this price and quality is forthcoming. Are you listening 'Creative Sparks'?



Java Jim

Boulder Dash

BOULDER DASH
Review by Barny Pilkinton.

This is a program from FIRST STAR Software, the company that brought you the award winning Astro Chase. Boulder Dash is of the same high standard. But enough of this, what's it all about you ask? Well . . .

The introductory title page is nothing special, but then several programs I have seen have only had the title page in their favour. The introduction music makes up for this by being fairly catchy. Pressing START takes you to the following options: 1 or 2 Players, 1 or 2 Joysticks, Starting Cave and Starting Level. The later 2 options are selected by Joystick movement and the first 2 options by, what else? - the Option button.

Now the object of the game is to move an ant-like creature around a very smooth scrolling screen collecting Diamonds, the number to be collected in each cave being displayed at the top of the screen. As you move Rockford - yes his name is Rockford - around he automatically digs tunnels to get to the Diamonds. But care must be taken when doing this as there are quite a lot of Rocks just waiting to be dislodged to crush poor Rockford and lose him one of his lives. Taking too long to pick your route through the caves results in your wasting your allotted time which is being counted off at the top of the screen, and this tickled me, Rockford tapping his foot impatiently whilst standing, arms folded, waiting to move on. Once you have collected enough Diamonds you will hear an explosion which lets you know that your exit from that cave is now obtainable, and all you have to do is find it.

Moving on to the higher caves brings not only different cave formations, but Fireflies, Butterflies, Dreaded Ameoba and underground fluids which fill in your tunnels. Rockford, however, has the grace of an extra life for each 500 points scored.

This game sports good graphics and a challenge, whilst not being too difficult for the whole family to enjoy and laugh over, for it certainly is amusing and one of my favourites.

BRUCE LEE
Review by Barny Pilkinton.

When first booting up 'Bruce Lee' you are confronted by what can only be described as a very tidy title page sporting an excellent likeness of the man himself. The musical score finally dies away, and I say finally, because it does become long-winded once the novelty has worn off and there seems to be no way of skipping the intro. Once you get to the main game you are offered several play options. These being: 1 player vs computer, 2 player vs computer and finally, and in my opinion the most exciting, Player vs Player.

The object of the game is to negotiate 20 screens containing permutations of flaming bushes and energy traps while collecting lanterns. Whilst working from room to room, to finally meet and destroy and evil wizard, you are harassed by a computer controlled Ninja called 'The Sword', this foe is controlled by the computer on all options, and a Sumo wrestler called a 'Green Yamu'. Both the Ninja and Sumo wrestler are out to foil Bruce Lee in his quest, the Ninja by flooring Bruce with his sword and the Sumo wrestler with both drop kicks and Karate style chops, the latter skills being available to Bruce Lee, who also has the advantage of being able to duck. All movement is controlled via the Joystick, with the 2nd player controlling the Sumo wrestler in the Player vs Player mode, which is otherwise controlled by the computer.

To assist in moving around the screens, Bruce can also climb trellises and leap gaps. Bruce loses a "fall" each time he is knocked out by either the Ninja or the Sumo wrestler. He starts off with 5 "Falls" and gains one for each 40,000 points. Should you complete the 20 rooms, it is implied in the manual that the sequence re-starts, but is much harder. This point you will have to try for yourself as I can only clear about eight screens.

Altogether, 'Bruce Lee' is a very good game from DATASOFT, in both graphic representation and playability. Perhaps a little expensive but still worth the money as it is not one to get bored with easily.

DID YOU KNOW

Using the Comma

When the ATARI computer's printer handler was written, it was assumed that most people would be using the 40 column printer, and consequently some of you are experiencing problems using 80 column printers. If you use two successive LPRINT statements and place commas after each, you do not get the neat tabbing effect you should. To get around this you must OPEN the printer as a file, and use the command PRINT#1;A$,. This forces the operating system to use its general text handler before dumping it to the printer.

Two further points about the use of the comma, are firstly that to obtain neat columns on the screen you must first POKE 82,Ø to give a 40 column display. Secondly, the comma tab is not the same as the TAB key on the keyboard (or CHR$(127)). You can alter the spacing of the first characters of two strings printed consecutively and separated by a comma by altering location 201 ($C9 HEX). The default value is ten, but this can be changed to any reasonable value, and the interesting thing is that it affects printing to any device, screen printers, cassette drives or disk drives!

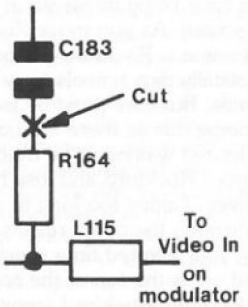
CORRIGENDA

Page 19. In the Label Maker listing line 6860 should read: NEXT L:FOR LOOP=1 TO 3:LPRINT:NEXT LOOP:RETURN

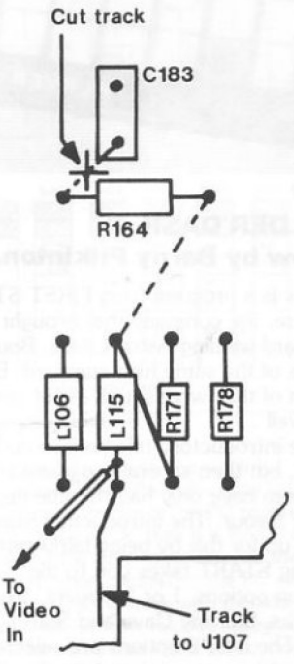
Page 20. In the list of Operators in Table 1, operator 35 (HEX 23) is not a blank space but should be the Power symbol (Shift *).

Page 23. In the De-tokeniser listing line 32010, because of the error in Table 1, the statement between = and * which shows a blank space should show the Power symbol (Shift *). Also in line 32013 the DATA should be PTRIG,STRIG not STICK,STRIG

Page 7. In the ATARI DIY article some additional information is required to fit the new modulator on an ATARI 400. Refering to Figure 2, the Video In from the modulator goes to L115 on the 400 PCB (the equivalent of L103 on the 800) and C183 does join to L115 but through R164 first. The circuit looks like this:



On the actual 400 Motherboard the changes would look like this:



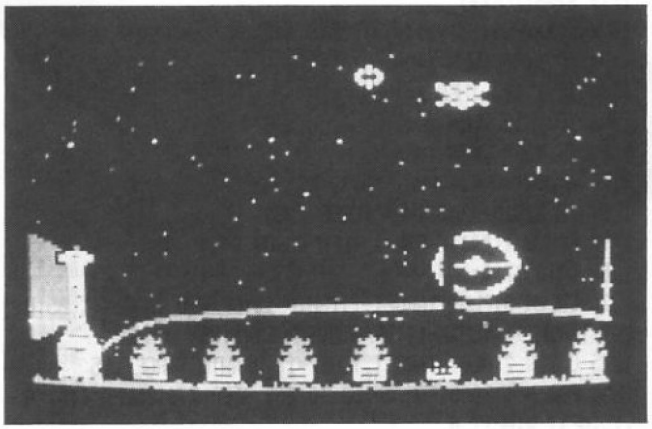
PLANETRON

Runs in 32K Cassette or 48K Disk

by Mark & Brian Christian — Wirral

You are in command of the Tower Laser Cannon, and it is up to you to defend the fuel dump from attack by the enemy spacecraft. As the Tiger Ship sweeps in for a low attack, use your joystick to target the enemy, then press your fire button for a beam of deadly energy. If your tower takes a direct hit, you can move over to a second tower to continue the fight, but this tower is not so powerful and it becomes harder to stop the enemy. The game finishes when all of the fuel dump is destroyed.

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



```
1000 CLR
1060 XC=14:GOSUB 2295
1070 GOSUB 2070
1080 GOTO 1500
1100 ON INT(RND(0)*3)+1 GOTO 1110,1120,1130
1110 YSHIP=YSHIP+3:GOTO 1130
1120 YSHIP=YSHIP-3
1130 IF YSHIP<15 THEN YSHIP=15
1140 IF YSHIP>50 THEN YSHIP=50
1150 IF DIR=1 THEN XSHIP=XSHIP+3
1160 IF DIR=2 THEN XSHIP=XSHIP-3
1165 IF SC>10000 THEN GOSUB 1222
1168 IF SC>20000 THEN GOSUB 1230
1169 IF SC>30000 THEN GOSUB 1236
1170 IF SHIP=1 THEN A=USR(MOVE,1,PMB,PM1,XSHIP,YSHIP,7)
1180 IF SHIP=2 THEN A=USR(MOVE,2,PMB,PM2,XSHIP,YSHIP,9)
1190 IF SHIP=3 THEN A=USR(MOVE,3,PMB,PM3,XSHIP,YSHIP,8)
1200 IF XSHIP<20 OR XSHIP>210 THEN GOSUB 1280
1210 IF XSHIP>TARGET+34 AND XSHIP<TARGET+48 THEN GOSUB 1
820
1220 RETURN
1222 IF DIR=1 THEN XSHIP=XSHIP+3.5
1224 IF DIR=2 THEN XSHIP=XSHIP-3.5
1225 RETURN
1230 IF DIR=1 THEN XSHIP=XSHIP+4
1232 IF DIR=2 THEN XSHIP=XSHIP-4
1235 RETURN
1236 IF DIR=1 THEN XSHIP=XSHIP+5
1237 IF DIR=2 THEN XSHIP=XSHIP-5
1238 RETURN
1240 NUM=INT(RND(0)*8)+1:TARGET=BUILDING(NUM)
1250 IF TARGET=999 THEN 1240
1260 RETURN
1280 SHIP=INT(RND(0)*3)+1
1290 DIR=INT(RND(0)*2)+1
1300 YSHIP=INT(RND(0)*35)+15
1310 IF DIR=1 THEN XSHIP=20
1320 IF DIR=2 THEN XSHIP=210
1330 IF SHIP=1 THEN POKE 53764,48:POKE 53765,70
1340 IF SHIP=2 THEN POKE 53764,42:POKE 53765,70
1350 IF SHIP=3 THEN POKE 53764,10:POKE 53765,70
1360 GOSUB 1240:RETURN
1380 POKE 53278,0:NM=INT(RND(0)*15)
1385 IF XC>15 AND NM>2 AND NM<8 THEN RETURN
1390 COLOR 3:PLOT XC,52:DRAWTO X-44,Y-12:FOR N=1 TO 50 S
TEP 5:SOUND 0,N,12,8:NEXT N:SOUND 1,0,0,0
1400 COLOR 0:PLOT XC,52:DRAWTO X-44,Y-12
1410 IF SHIP=1 THEN IF PEEK(53253)<>0 AND PEEK(53261)<>0
 THEN EXP=1:SD=40:SC=SC+500:POKE 53249,0:GOSUB 1280
1420 IF SHIP=2 THEN IF PEEK(53254)<>0 AND PEEK(53262)<>0
 THEN EXP=1:SD=40:SC=SC+100:POKE 53250,0:GOSUB 1280
1430 IF SHIP=3 THEN IF PEEK(53255)<>0 AND PEEK(53263)<>0
 THEN EXP=1:SD=40:SC=SC+300:POKE 53251,0:GOSUB 1280
1440 RETURN
1460 SD=SD+5:IF SD>100 THEN SOUND 1,0,0,0:EXP=0:RETURN
1470 SETCOLOR 4,0,14:SETCOLOR 4,3,4:SETCOLOR 4,0,0
1480 SOUND 1,SD,8,8:RETURN
1490 REM
1500 TRAP 1510:DIM BUILDING(8),XPOS(15),YPOS(15),PMMOV$(
100),P0$(7),P1$(7),P2$(9),P3$(8)
1510 X=125:Y=70:SC=0:HITS=0:MOVE=ADR(PMMOV$):PM0=ADR(P0$
):PM1=ADR(P1$):PM2=ADR(P2$):PM3=ADR(P3$)
1520 RESTORE 1530:FOR N=1 TO 8:READ A:BUILDING(N)=A:NEXT
 N
1530 DATA 14,34,54,74,94,114,134,154
1540 RESTORE 1550:FOR N=1 TO 15:READ A:XPOS(N)=A:READ A:
YPOS(N)=A:NEXT N
1550 DATA 0,0,0,0,0,0,0,0,3,3,3,-3,3,0,0,0,-3,3,-3,-3,-3
,0,0,0,0,3,0,-3,0,0
1560 RESTORE 1570:FOR N=1 TO 100:READ A:PMMOV$(N)=CHR$(A
):NEXT N
1570 DATA 216,104,104,104,133,213,104,24,105,2,133,206,1
04,133,205,104,133,204,104,133,203,104,104,133,208
1580 DATA 104,104,133,209,104,104,24,101,209,133,207,166
,213,240,16,165,205,24,105,128,133,205,165,206,105
1590 DATA 0,133,206,202,208,240,160,0,162,0,196,209,144,
19,196,207,176,15,132,212,138,168,177,203,164
1600 DATA 212,145,205,232,169,0,240,4,169,0,145,205,200,
192,128,208,224,166,213,165,208,157,0,208,96
1610 RESTORE 1650:FOR N=1 TO 7:READ A:P0$(N)=CHR$(A):NEX
T N
1620 FOR N=1 TO 7:READ A:P1$(N)=CHR$(A):NEXT N
1630 FOR N=1 TO 9:READ A:P2$(N)=CHR$(A):NEXT N
1640 FOR N=1 TO 8:READ A:P3$(N)=CHR$(A):NEXT N
1650 DATA 8,42,65,65,42,8,0,186,214,108,56,84,186,16,189
,219,153
1660 DATA 24,231,66,129,90,60,198,108,214,124,170,186,68
,56
1670 PMBASE=INT((PEEK(145)+3)/4)*4:POKE 54279,PMBASE:PMB
=PMBASE*256
```

```
1680 POKE 704,30:POKE 705,216:POKE 706,246:POKE 707,154
1690 POKE 623,1:POKE 53257,1:POKE 53258,1:POKE 53259,1:P
OKE 559,46:POKE 53277,3:GOSUB 1280
1710 ST=STICK(0)
1720 X=X+XPOS(ST)*2:Y=Y+YPOS(ST)*2
1730 IF X<48 THEN X=48
1740 IF X>198 THEN X=198
1750 IF Y<12 THEN Y=12
1760 IF Y>50 THEN Y=50
1770 A=USR(MOVE,0,PMB,PM0,X,Y,7)
1780 IF STRIG(0)=0 THEN GOSUB 1380
1790 GOSUB 1100:IF EXP=1 THEN GOSUB 1460
1800 GOTO 1710
1820 COLOR 3:PLOT TARGET-1,YSHIP-10:DRAWTO TARGET-1,85:P
LOT TARGET+1,YSHIP-10:DRAWTO TARGET+1,85
1825 COLOR 2:PLOT TARGET,YSHIP-10:DRAWTO TARGET,85
1830 FOR N=1 TO 5:FOR A=150 TO 250 STEP 7.6:SOUND 0,A,14
,15:NEXT A:NEXT N
1840 COLOR 0:PLOT TARGET-1,YSHIP-10:DRAWTO TARGET-1,85:P
LOT TARGET+1,YSHIP-10:DRAWTO TARGET+1,85
1845 COLOR 0:PLOT TARGET,YSHIP-10:DRAWTO TARGET,85
1850 SCR=PEEK(560)+PEEK(561)*256:B=PEEK(SCR+4)
1860 FOR N=1 TO 50:POKE SCR+4,B+PEEK(53770)/200:SOUND 0,
N,0,8:NEXT N
1865 IF TARGET=14 THEN XC=158:COLOR 0:FOR I=8 TO 19:PLOT
 I,55:DRAWTO I,94-RND(0)*5:NEXT I:GOSUB 2950
1870 FOR N=TARGET-7 TO TARGET+5:COLOR 0:PLOT N,80:DRAWTO
 N,91-RND(0)*5:SOUND 0,N,0,8:NEXT N
1880 FOR N=1 TO 50:SETCOLOR 4,0,14:SOUND 0,N,0,8:SETCOLO
R 4,0,0:NEXT N
1890 SOUND 0,0,0,0:POKE SCR+4,B
1900 HITS=HITS+1
1910 IF HITS=8 THEN SOUND 2,0,0,0:FOR N=1 TO 200:NEXT N:
FOR N=53248 TO 53251:POKE N,0:NEXT N:GOTO 1950
1920 BUILDING(NUM)=999:GOSUB 1240
1930 RETURN
1950 GRAPHICS 17:DL=PEEK(560)+PEEK(561)*256+4:POKE DL-1,
71:POKE DL+2,7
1960 POKE DL+3,7:POKE DL+4,7:POKE DL+11,7:POKE DL+12,7:P
OKE DL+13,7:POKE DL+14,7
1970 POKE DL+15,65:POKE DL+16,PEEK(560):POKE DL+17,PEEK(
561)
1980 SETCOLOR 0,4,4:SETCOLOR 1,3,4:SETCOLOR 2,RND(0)*15,
10:SETCOLOR 3,2,6:TRAP 2030:IF SC>HSC THEN HSC=SC
1990 POKE 87,2:POSITION 5,1:? #6;"game over":POKE 87,1:P
OSITION 3,7:? #6;"SCORE - ";SC
2000 POSITION 1,9:? #6;"HIGH SCORE - ";HSC:POSITION 7,11
:? #6;"press":POSITION 5,13:? #6;" TRIGGER "
2010 DIM X$(70):X$=",,,all,fuel,dumps,gone,,planet,earth
,doomed,,invasion imminent,,,"
2020 DIM A$(LEN(X$)),C$(LEN(X$)):A$=X$
2030 REM
2032 RESTORE 2042
2035 READ A:IF A=255 THEN 2032
2040 READ B:SOUND 0,B,10,8:SOUND 1,B-1,10,8:FOR N=1 TO A
*4:NEXT N:SOUND 0,0,0,0:SOUND 1,0,0,0
2041 IF STRIG(0)=0 THEN SOUND 3,0,0,0:GOTO 2050
2042 POSITION 0,4:? #6;A$(1,20):C$=A$(2):C$(LEN(C$)+1)=A
$:A$=C$
2043 GOTO 2035
2044 DATA 2,228,2,228,4,180,2,212,2,240,4,180,2,212,2,24
0,4,180,2,212,2,240,4,240,1,240,1,240,6,240
2045 DATA 2,254,4,240,2,254,4,240,2,254,4,240,2,240,2,24
0,1,240,8,164,255
2050 FOR N=0 TO 3:SOUND N,0,0,0:NEXT N:GOTO 1060
2070 GRAPHICS 7+16:SETCOLOR 0,0,8:SETCOLOR 1,3,4:SETCOLO
R 2,12,4:GOSUB 2950
2074 RESTORE 2074:COLOR 1:FOR B=1 TO 37:READ R,S,T:PLOT
S,R:DRAWTO T,R:NEXT B
2075 DATA 50,118,124,51,116,126,52,114,117,52,124,128,53
,113,115,53,126,130,54,112,114,54,128,131
2076 DATA 55,112,113,55,129,131,56,111,112,56,130,132,57
,111,112,57,130,132,58,110,111,58,131,133
2077 DATA 59,110,133,60,110,111,60,132,133,61,111,112,61
,131,132,62,111,112,62,131,132
2078 DATA 63,112,113,63,130,131,64,112,114,64,129,131,65
,113,116,65,127,130,66,115,118,66,125,129
2079 DATA 67,117,127,68,119,125,57,121,122,58,119,123,60
,119,123,61,120,121
2080 COLOR 1:FOR N=6 TO 94:POKE DL+N,141:NEXT N
2081 FOR J=0 TO 3:READ A:POKE 1536+J,A:NEXT J:COLOR 3:PO
KE 512,0:POKE 513,6:POKE 54286,192
2082 DATA 142,24,208,64
2083 COLOR 3:FOR N=1 TO 150:PLOT RND(0)*159,RND(0)*90:NE
XT N
2088 COLOR 1:PLOT 158,53:DRAWTO 158,76:PLOT 157,61:PLOT
157,66:PLOT 157,71:PLOT 157,76
2089 PLOT 159,61:PLOT 159,66:PLOT 159,71:PLOT 159,76:GOS
UB 2900
2090 COLOR 1:FOR I=8 TO 19:PLOT I,77:DRAWTO I,94:PLOT I,
55:DRAWTO I,58:NEXT I
2095 PLOT 12,59:DRAWTO 12,76:PLOT 13,59:DRAWTO 13,76:PLO
T 14,59:DRAWTO 14,76:PLOT 15,59:DRAWTO 15,76
2110 COLOR 2:PLOT 13,53:DRAWTO 13,54:DRAWTO 14,54:DRAWTO
 14,53:PLOT 12,87:DRAWTO 15,87:PLOT 13,89:PLOT 14,89
2115 PLOT 12,91:PLOT 15,91:PLOT 13,85:PLOT 14,85
2120 COLOR 3:PLOT 13,56:PLOT 14,56:PLOT 13,82:PLOT 13,78
:PLOT 14,78:PLOT 14,77:PLOT 10,82
2125 PLOT 11,82:PLOT 16,82:PLOT 17,82:PLOT 10,91:PLOT 11
,91:PLOT 16,91:PLOT 17,91
2130 COLOR 0:PLOT 8,90:PLOT 8,91:PLOT 19,90:PLOT 10,93:P
LOT 11,93:PLOT 19,91:PLOT 13,94
2135 PLOT 13,93:PLOT 14,94:PLOT 14,93:PLOT 16,93:PLOT 17
,93:PLOT 10,84:DRAWTO 12,84
2140 PLOT 13,84:DRAWTO 15,84:PLOT 11,80:DRAWTO 16,80
2150 PLOT 8,82:DRAWTO 8,77:DRAWTO 10,77:DRAWTO 10,78:DRA
WTO 9,78:DRAWTO 9,80
2160 PLOT 19,82:DRAWTO 19,77:DRAWTO 17,77:DRAWTO 17,78:D
RAWTO 18,78:DRAWTO 18,80
2170 PLOT 14,74:PLOT 13,71:PLOT 14,68:PLOT 13,65
2180 PLOT 10,55:DRAWTO 8,55:DRAWTO 8,58:DRAWTO 10,58:PLO
T 17,55:DRAWTO 19,55:DRAWTO 19,58:DRAWTO 17,58
2199 COLOR 3:FOR N=0 TO 159:PLOT N,95:DRAWTO N,95-(2*RND
(0)):NEXT N
2200 FOR N=28 TO 158 STEP 20
2210 COLOR 1:FOR I=N+1 TO N+9:PLOT I,86:DRAWTO I,92:NEXT
 I
2220 COLOR 2:FOR I=N TO N+10:PLOT I,79:DRAWTO I,87:NEXT
I
2230 COLOR 0:PLOT N+2,85:DRAWTO N,85:DRAWTO N,79:DRAWTO
N+4,79:PLOT N+6,79:DRAWTO N+10,79:DRAWTO N+10,85:DRAWTO
N+8,85
2240 PLOT N+3,80:PLOT N+7,80:PLOT N+2,83:PLOT N+8,83:PLO
T N+1,86:PLOT N+8,86:PLOT N+3,93:PLOT N+7,93
```

```
2250 PLOT N+1,80:DRAWTO N+1,82:DRAWTO N+3,82:PLOT N+9,80
:DRAWTO N+9,82:DRAWTO N+7,82
2260 PLOT N+3,89:DRAWTO N+7,89:PLOT N+3,91:DRAWTO N+7,91
:PLOT N+5,93:DRAWTO N+5,95
2262 COLOR 2:PLOT N+3,94:DRAWTO N+7,94
2290 NEXT N:RETURN
2295 GRAPHICS 7+16:SETCOLOR 0,3,4:SETCOLOR 2,7,4:SETCOLO
R 1,0,12
2296 COLOR 3:PLOT 0,0:DRAWTO 159,0:DRAWTO 159,80:DRAWTO
0,80:DRAWTO 0,0
2297 DL=PEEK(560)+PEEK(561)*256+4
2298 POKE DL+89,6:POKE DL+90,6:POKE DL+91,65:POKE DL+92,
PEEK(560):POKE DL+93,PEEK(561)
2303 COLOR 2:FOR N=1 TO 50:PLOT RND(0)*159,RND(0)*80:NEX
T N
2304 COLOR 2:FOR I=5 TO 15:PLOT 68,I:DRAWTO 149,I:NEXT I
:FOR I=16 TO 19:PLOT 74,I:DRAWTO 143,I:NEXT I
2305 FOR I=20 TO 24:PLOT 74,I:DRAWTO 143,I:NEXT I:FOR I=
25 TO 28:PLOT 86,I:DRAWTO 131,I:NEXT I
2310 FOR I=29 TO 38:PLOT 82,I:DRAWTO 135,I:NEXT I:FOR I=
39 TO 43:PLOT 97,I:DRAWTO 120,I:NEXT I
2315 COLOR 0:PLOT 68,13:DRAWTO 82,13:DRAWTO 88,16:DRAWTO
 98,16:DRAWTO 106,20:DRAWTO 111,20:DRAWTO 119,16:DRAWTO
129,16:DRAWTO 135,13:DRAWTO 149,13
2320 PLOT 84,5:DRAWTO 133,5:PLOT 86,6:DRAWTO 131,6:PLOT
88,7:DRAWTO 129,7
2325 PLOT 100,8:DRAWTO 117,8:PLOT 102,9:DRAWTO 115,9:PLO
T 104,10:DRAWTO 113,10:PLOT 106,11:DRAWTO 111,11
2330 PLOT 68,13:DRAWTO 68,15:PLOT 69,14:DRAWTO 69,15:PLO
T 70,15:PLOT 149,13:DRAWTO 149,15:PLOT 148,14:DRAWTO 148
,15:PLOT 147,15
2335 PLOT 78,16:DRAWTO 78,19:PLOT 79,17:DRAWTO 79,19:PLO
T 80,18:DRAWTO 80,19:PLOT 81,19
2340 PLOT 139,16:DRAWTO 139,19:PLOT 138,17:DRAWTO 138,19
:PLOT 137,18:DRAWTO 137,19:PLOT 136,19
2345 PLOT 74,21:DRAWTO 74,24:PLOT 75,22:DRAWTO 75,24:PLO
T 76,23:DRAWTO 76,24:PLOT 77,24
2350 PLOT 143,21:DRAWTO 143,24:PLOT 142,22:DRAWTO 142,24
:PLOT 141,23:DRAWTO 141,24:PLOT 140,24
2355 PLOT 86,26:DRAWTO 86,28:PLOT 87,27:DRAWTO 87,28:PLO
T 88,28:PLOT 131,26:DRAWTO 131,28:PLOT 130,27:DRAWTO 130
,28:PLOT 129,28
2360 PLOT 82,30:DRAWTO 82,38:PLOT 83,31:DRAWTO 83,38:PLO
T 84,32:DRAWTO 84,38:PLOT 85,33:DRAWTO 85,38
2365 PLOT 86,34:DRAWTO 86,38:PLOT 87,35:DRAWTO 87,38:PLO
T 88,36:DRAWTO 88,38:PLOT 89,37:DRAWTO 89,38:PLOT 90,38
2370 PLOT 135,30:DRAWTO 135,38:PLOT 134,31:DRAWTO 134,38
:PLOT 133,32:DRAWTO 133,38:PLOT 132,33:DRAWTO 132,38
2371 PLOT 131,34:DRAWTO 131,38:PLOT 130,35:DRAWTO 130,38
:PLOT 129,36:DRAWTO 129,38:PLOT 128,37:DRAWTO 128,38:PLO
T 127,38
2373 PLOT 87,18:DRAWTO 88,18:PLOT 88,19:DRAWTO 100,19
2374 PLOT 131,18:DRAWTO 130,18:PLOT 130,19:DRAWTO 117,19
2375 PLOT 117,38:DRAWTO 112,33:DRAWTO 112,28:DRAWTO 105,
28:DRAWTO 105,33:DRAWTO 100,38:PLOT 105,27:DRAWTO 108,20
:DRAWTO 109,20:DRAWTO 112,27
2376 FOR I=70 TO 78:PLOT I,16:DRAWTO I,19:NEXT I:FOR I=1
40 TO 148:PLOT I,16:DRAWTO I,19:NEXT I
2377 FOR I=91 TO 96:PLOT I,39:DRAWTO I,43:NEXT I:FOR I=1
21 TO 126:PLOT I,39:DRAWTO I,43:NEXT I
2378 PLOT 97,40:DRAWTO 97,43:PLOT 98,41:DRAWTO 98,43:PLO
T 99,42:DRAWTO 99,43:PLOT 100,43
2379 PLOT 120,40:DRAWTO 120,43:PLOT 119,41:DRAWTO 119,43
:PLOT 118,42:DRAWTO 118,43:PLOT 117,43
2380 COLOR 1:PLOT 71,7:DRAWTO 80,7:PLOT 71,8:DRAWTO 80,8
:PLOT 71,9:DRAWTO 80,9:PLOT 71,10:DRAWTO 80,10
2385 COLOR 3:PLOT 84,7:DRAWTO 84,12:PLOT 85,7:DRAWTO 85,
12:PLOT 86,8:DRAWTO 86,13:PLOT 87,8:DRAWTO 87,13
2390 PLOT 100,11:DRAWTO 100,16:PLOT 101,11:DRAWTO 101,16
:PLOT 102,12:DRAWTO 102,17
2395 PLOT 103,12:DRAWTO 103,17:PLOT 104,13:DRAWTO 104,18
:PLOT 105,13:DRAWTO 105,18
2400 PLOT 112,13:DRAWTO 112,18:PLOT 113,13:DRAWTO 113,18
:PLOT 114,12:DRAWTO 114,17
2405 PLOT 115,12:DRAWTO 115,17:PLOT 116,11:DRAWTO 116,16
:PLOT 117,11:DRAWTO 117,16
2410 PLOT 130,8:DRAWTO 130,13:PLOT 131,8:DRAWTO 131,13:P
LOT 132,7:DRAWTO 132,12:PLOT 133,7:DRAWTO 133,12
2415 COLOR 1:PLOT 137,7:DRAWTO 146,7:PLOT 137,8:DRAWTO 1
46,8:PLOT 137,9:DRAWTO 146,9:PLOT 137,10:DRAWTO 146,10
2420 PLOT 90,21:DRAWTO 101,21:PLOT 91,22:DRAWTO 102,22:P
LOT 92,23:DRAWTO 103,23
2425 PLOT 116,21:DRAWTO 127,21:PLOT 115,22:DRAWTO 126,22
:PLOT 114,23:DRAWTO 125,23
2427 COLOR 3:PLOT 108,26:DRAWTO 109,26:PLOT 107,27:DRAWT
O 110,27
2430 PLOT 92,29:DRAWTO 97,35:PLOT 93,29:DRAWTO 98,35:PLO
T 94,29:DRAWTO 99,35:PLOT 95,29:DRAWTO 100,35
2435 PLOT 96,29:DRAWTO 101,35:PLOT 97,29:DRAWTO 102,35:P
LOT 106,29:DRAWTO 111,29:PLOT 107,30:DRAWTO 110,30
2440 PLOT 108,31:DRAWTO 109,31:PLOT 120,29:DRAWTO 115,35
:PLOT 121,29:DRAWTO 116,35:PLOT 122,29:DRAWTO 117,35
2445 PLOT 123,29:DRAWTO 118,35:PLOT 124,29:DRAWTO 119,35
:PLOT 125,29:DRAWTO 120,35:COLOR 1:PLOT 102,37:DRAWTO 11
5,37
2450 PLOT 103,38:DRAWTO 114,38:PLOT 104,39:DRAWTO 113,39
:PLOT 105,40:DRAWTO 112,40:PLOT 106,41:DRAWTO 111,41
2460 COLOR 2:PLOT 104,69:DRAWTO 113,69:PLOT 96,70:DRAWTO
 121,70:PLOT 91,71:DRAWTO 126,71:PLOT 87,72:DRAWTO 130,7
2
2461 PLOT 84,73:DRAWTO 133,73:PLOT 81,74:DRAWTO 136,74:P
LOT 79,75:DRAWTO 138,75:PLOT 77,76:DRAWTO 140,76
2462 PLOT 76,77:DRAWTO 141,77:PLOT 75,78:DRAWTO 142,78:P
LOT 74,79:DRAWTO 143,79
2465 COLOR 1:FOR VZ=84 TO 143 STEP 5:PLOT 109,78:DRAWTO
VZ,78-RND(0)*16:NEXT VZ
2466 COLOR 0:FOR VZ=90 TO 142 STEP 7:PLOT 109,78:DRAWTO
VZ,79-RND(0)*6:NEXT VZ
2468 COLOR 3:PLOT 8,52:DRAWTO 8,62:PLOT 9,52:DRAWTO 9,62
:PLOT 10,52:DRAWTO 14,52:PLOT 10,57:DRAWTO 14,57
2470 PLOT 14,53:DRAWTO 16,53:PLOT 14,56:DRAWTO 16,56:PLO
T 16,54:DRAWTO 16,55:PLOT 17,54:DRAWTO 17,55
2472 PLOT 20,52:DRAWTO 20,61:PLOT 21,52:DRAWTO 21,61:DRA
WTO 27,61
2474 PLOT 30,55:DRAWTO 30,61:PLOT 31,55:DRAWTO 31,61:PLO
T 38,55:DRAWTO 38,61:PLOT 39,55:DRAWTO 39,61
2476 PLOT 31,54:DRAWTO 33,52:DRAWTO 36,52:DRAWTO 38,54:P
LOT 32,54:DRAWTO 33,53:PLOT 36,53:DRAWTO 37,54
2477 PLOT 32,58:DRAWTO 37,58
2478 PLOT 42,52:DRAWTO 42,61:PLOT 43,52:DRAWTO 43,61:PLO
T 50,52:DRAWTO 50,61:PLOT 51,52:DRAWTO 51,61
2480 PLOT 44,52:DRAWTO 44,54:PLOT 45,54:DRAWTO 45,56:PLO
T 46,55:DRAWTO 46,57:PLOT 47,56:DRAWTO 47,58
```

```
2482 PLOT 48,57:DRAWTO 48,59:PLOT 49,58:DRAWTO 49,61
2484 PLOT 54,52:DRAWTO 54,61:PLOT 55,52:DRAWTO 55,61:PLO
T 56,52:DRAWTO 63,52:PLOT 56,61:DRAWTO 63,61
2486 PLOT 56,57:DRAWTO 59,57
2488 PLOT 66,52:DRAWTO 75,52:PLOT 70,52:DRAWTO 70,61:PLO
T 71,52:DRAWTO 71,61
2490 PLOT 78,52:DRAWTO 78,61:PLOT 79,52:DRAWTO 79,61:PLO
T 80,52:DRAWTO 84,52:PLOT 80,57:DRAWTO 84,57
2492 PLOT 84,53:DRAWTO 86,53:PLOT 84,56:DRAWTO 86,56:PLO
T 86,54:DRAWTO 86,55:PLOT 87,54:DRAWTO 87,55
2494 PLOT 82,58:DRAWTO 85,61:PLOT 83,58:DRAWTO 86,61
2496 PLOT 90,55:DRAWTO 90,58:DRAWTO 93,61:DRAWTO 96,61:D
RAWTO 99,58:DRAWTO 99,55:DRAWTO 96,52:DRAWTO 93,52:DRAWT
O 90,55
2498 PLOT 93,53:DRAWTO 91,55:DRAWTO 91,58:DRAWTO 93,60:P
LOT 96,60:DRAWTO 98,58:DRAWTO 98,55:DRAWTO 96,53
2500 PLOT 102,52:DRAWTO 102,61:PLOT 103,52:DRAWTO 103,61
:PLOT 110,52:DRAWTO 110,61:PLOT 111,52:DRAWTO 111,61
2502 PLOT 104,52:DRAWTO 104,54:PLOT 105,54:DRAWTO 105,56
:PLOT 106,55:DRAWTO 106,57:PLOT 107,56:DRAWTO 107,58
2504 PLOT 108,57:DRAWTO 108,59:PLOT 109,58:DRAWTO 109,61
2520 TPS=PEEK(88)+PEEK(89)*256:POKE 87,1:TPS=TPS+3521:PO
KE 88,TPS-(INT(TPS/256)*256):POKE 89,INT(TPS/256)
2530 POSITION 0,0:? #6;"BY M.&.B.CHRISTIAN!"
2540 POSITION 4,1:? #6;"PRESS FIRE "
2550 RESTORE 2600:SOUND 2,215,10,10
2560 READ DUR:SETCOLOR 0,DUR+2,12:IF DUR=254 THEN 2550
2570 READ TONE:SOUND 0,TONE,10,8:SOUND 1,TONE-1,10,8:FOR
 N=1 TO DUR*12:NEXT N:SOUND 0,0,0,0:SOUND 1,0,0,0
2580 IF STRIG(0)=0 THEN SOUND 2,0,0,0:GOTO 2610
2590 GOTO 2560
2600 DATA 7,81,1,91,1,60,3,53,3,72,3,81,1,91,1,81,5,91,5
,96,1,108,1,96,5,108,5,144,5,193,5,204,1,203,1,225,5
2601 DATA 182,5,162,254
2610 GRAPHICS 2+16:SETCOLOR 4,10,0:SETCOLOR 0,3,4
2620 POSITION 3,3:? #6;"DO YOU REQUIRE":POSITION 3,4:? #
6;"THE COMPUTER":POSITION 2,5:? #6;"FOR READ-OUT...?"
2630 POSITION 6,7:? #6;"(Y OR N)"
2640 OPEN #1,4,0,"K":GET #1,A:CLOSE #1
2650 IF A=89 THEN 2680
2660 IF A=78 THEN RETURN
2670 GOTO 2640
2680 GRAPHICS 0:SETCOLOR 2,9,0:POKE 82,1:POKE 83,38:POKE
 752,1
2690 POSITION 8,3:? " - COMPUTER READ-OUT - ":? :?
2700 ? "  Many eons ago your ancestors  fought off the a
lien hordes from A distant  and unearthly race.";
2710 ? " These creatures  were out to destroy earths on
ly supplyof fuel.Many battles were won and lost";
2720 ? "but eventually they retreated and left for deep
space,never to return again.  "
2730 ? " BUT suddenly on your scanners fleets of highly
sophisticated craft appear  ";
2740 ? "from where the aliens had dissappeared.Now they
are stronger due to complete robotization.....";
2750 ? :? :? "    Press a key for next page    ":OPEN #1
,4,0,"K:":GET #1,A:CLOSE #1
2760 ? CHR$(125):? :? :? "  There are three main types o
f craft which you must remember.The first are the";
2770 ? " TIGER CRUISERS.....these have the appearance o
f a tiger are quite large in bulk. ";
2780 ? "You will find them slow but powerful craft.The n
ext are the SKULL DESTROYERS.....,which";
2790 ? " are vicious in their destructive mode.You will
find  them evasive and fast.Finally the most";
2800 ? " fearsome of all the SINISTER MASK    CRAFT.....
these bare the smiling cruel face of their ruler";
2810 ? ". These are worst of all...":? " So man your def
ence post and protect it and the fuel cities. If your ";
2820 ? "tower is destroyed you will fire from a less abl
e tower.....,GOOD LUCK!...."
2825 ? :? "  PRESS ANY KEY TO BEGIN YOUR MISSION "
2830 OPEN #1,4,0,"K:":GET #1,A:CLOSE #1
2840 RETURN
2900 COLOR 3:PLOT 15,89:DRAWTO 17,85:DRAWTO 25,79:DRAWTO
 33,75:DRAWTO 41,73:DRAWTO 78,70
2910 PLOT 78,70:DRAWTO 112,70:DRAWTO 149,73:DRAWTO 157,7
5
2920 COLOR 2:PLOT 15,90:DRAWTO 17,86:DRAWTO 25,80:DRAWTO
 33,76:DRAWTO 41,74:DRAWTO 78,71
2930 PLOT 78,71:DRAWTO 112,71:DRAWTO 149,74:DRAWTO 157,7
6
2940 RETURN
2950 RESTORE 2950:COLOR 3:FOR A=1 TO 16:READ X,Y,Z:PLOT
X,Y:DRAWTO X,Z:NEXT A
2960 DATA 0,50,81,1,50,81,2,50,81,3,51,80,4,51,80,5,52,7
9,6,52,79,7,53,78,8,53,78
2970 DATA 9,54,77,10,55,76,11,56,75,12,57,74,13,58,73,14
,60,71,15,63,68
2980 RETURN
```

THE U.K.

ATARI

COMPUTER OWNERS CLUB

INDEPENDENT USER GROUP

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