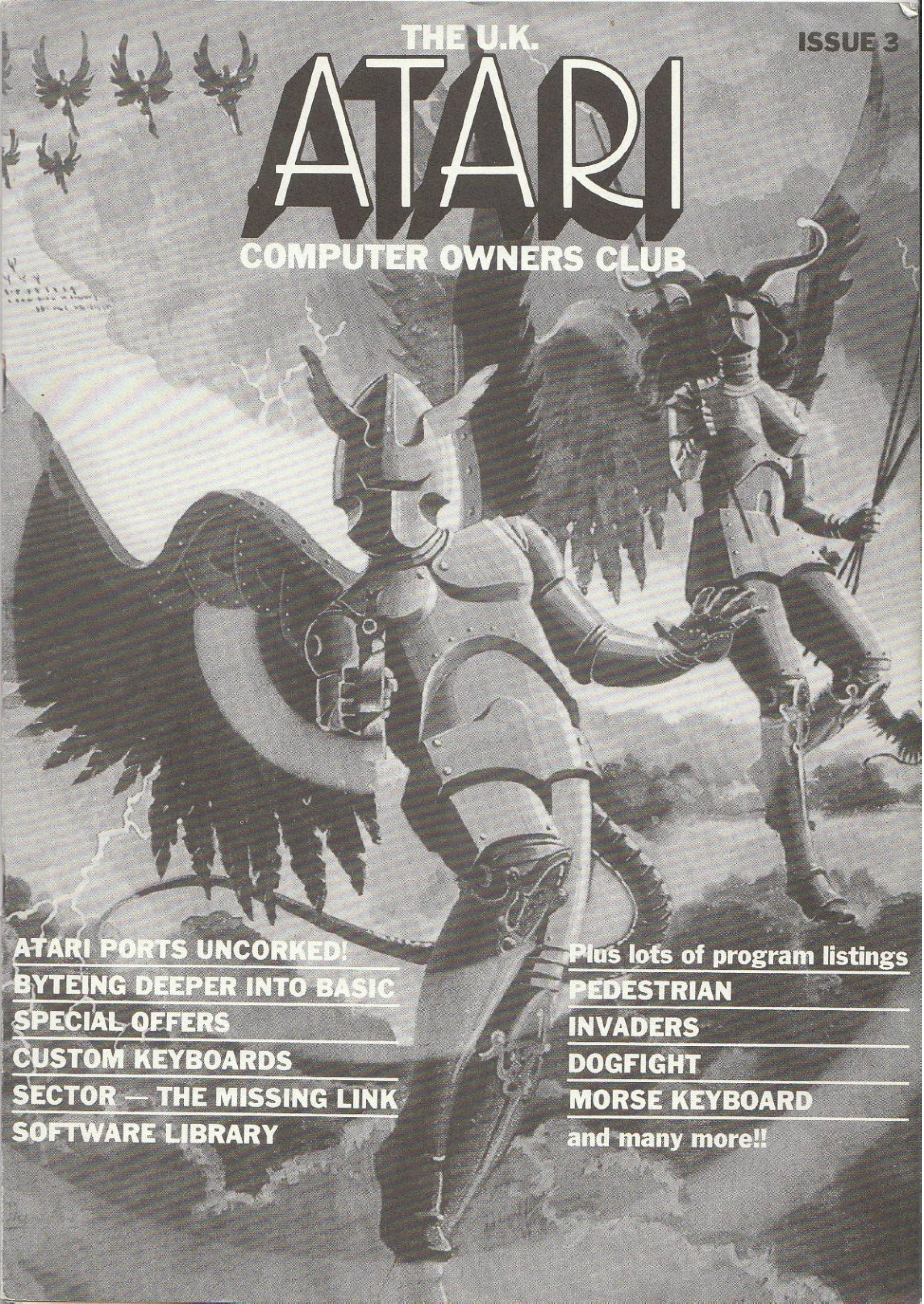
ISSUE 3

THE U.K. ATARI COMPUTER OWNERS CLUB



ATARI PORTS UNCORKED!

BYTEING DEEPER INTO BASIC

SPECIAL OFFERS

CUSTOM KEYBOARDS

SECTOR — THE MISSING LINK

SOFTWARE LIBRARY

Plus lots of program listings

PEDESTRIAN

INVADERS

DOGFIGHT

MORSE KEYBOARD

and many more!!

THE U.K.
ATARI
COMPUTER OWNERS CLUB

CONTENTS

DIAL UP YOUR CLUB

We proudly present issue three of your ATARI newsletter. As you have probably noticed by now, the newsletter has taken another leap forward in its presentation. This improvement has been possible because of our swelling membership. In fact, we now have over 1,200 members, and this number is increasing daily. We hope you find this issue as interesting and informative as you seem to have found the first two.

In this issue we have articles on the ATARI joystick ports, custom keyboards, an in depth feature on ATARI BASIC, readers letters and comments, and of course the software library listing. Also included are several complete program listings for you to type in.

CREDITS

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

From 1st June, you will be able to "dial up" to the U.K. ATARI Computer Owners Club. On the MAPLIN Computer, here at Hadleigh, Essex, a page has been allocated to us and we are hoping to present extra information, articles and features.

Also, later on we may be able to present program listings.

Of course you will have to own (or borrow) a modem which is European Standard (CCITT) and transmits at 300 baud to communicate with the MAPLIN Digital Equipment (DEC) PDP11/70 computer. You will also require an Interface 850 unit. The number to ring is 0702 552941. You will be presented with the MAPTEL menu screen, and one of the many options available will be "ATARI USERS GROUP". Select this option to receive the information that we have provided. We hope to update/change this information at regular intervals. So if you do not have a modem, rush out and buy one! P.S. MAPLIN sell a suitable one for approximately £40, but you have to build it yourself.

Finally, on the envelope in which this newsletter was sent, you will find your MEMBERSHIP number. Please use this number in any correspondence to us.

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

BYTEING DEEPER INTO BASIC

It is a sad fact that very few ATARI computer owners really know how to use their BASIC language to its full potential. Although BASIC will never equal the speed and efficiency of machine code, it does allow many people to write business, educational, general utility and moderate graphics programs with considerable speed and fluency.

Whilst there have been many articles on how to use the many graphics and other hardware functions of the machine, there has not been one to explain in a more detailed manner the normal BASIC commands. We intend in this article to take many of the commands and describe them in greater detail.

CONT

This seems like a very straight forward command with no complications, but how very wrong! If you stop a program in execution, and then resume with a CONT command then it will carry on at the start of the next line number. Great, so what happened to the other commands on a multiple statement line? Take a look at and try the following:

```
10 DIM A$(1):A$=" "
20 A=A+1:B=B+1:C=C+1:FOR LOOP=0 TO
100:NEXT LOOP:D=D+1:E=E+1:F=F+1
30 PRINT A;A$;B;A$;C;A$;D;A$;E;A$;F
40 GOTO 20
```

The program increments equally and then prints out six variables. RUN the program for a few seconds and then press BREAK. All the numbers on each line will of course be equal. Now type CONT and press BREAK again. You may now find that the numbers may not be equal on each line. If not, then try it again! This is due to it being stopped halfway through line 20 and although the break function allows the BASIC to complete its current command, it stops at the next colon, whereas the CONT instruction causes the BASIC to continue on the next line number. The FOR/NEXT loop is to increase the possibility of the program being stopped at this line. This is a problem which you should be aware of whilst debugging etc.

ON/GOTO & ON/GOSUB

These present a fairly powerful means of jumping to a routine depending on the value of a variable. Note the following example:

```
10 X=3
20 ON X GOTO 401,260,311,230
30 STOP
230 REM IF X=4 THEN IT JUMPS HERE
260 REM IF X=2 THEN IT JUMPS HERE
311 REM IF X=3 THEN IT JUMPS HERE
401 REM IF X=1 THEN IT JUMPS HERE
```

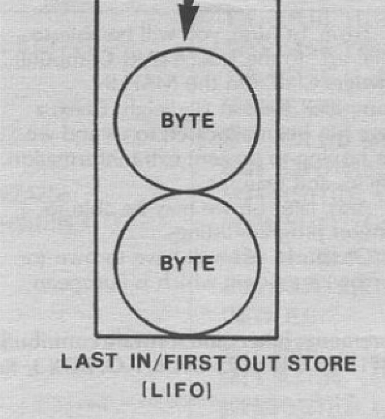
A quick look at the program will show that the routine chosen in the ON X GOTO statement is nothing to do with the numerical values of the GOTO line numbers, but it is the actual position that is important, i.e. if X=3 then it will choose the third GOTO line number (311). You can also use the GOSUB command in place of the GOTO and with ATARI BASIC a variable can be used in place of actual line numbers, e.g.:

```
10 A=100:X=1:B=200:H=1200:Y=1010
20 GOSUB A
30 ON X GOTO B,H,Y
100 REM IT JUMPS HERE FIRST
110 RETURN
200 REM IT JUMPS HERE SECOND
210 STOP
1010 REM GOES HERE IF X=3
1200 REM GOES HERE IF X=2
```

In fact, a variable can be used in place of a real line number where ever a GOSUB or GOTO are used.

POP

Understanding the POP command requires knowledge of the meaning of a 'stack'. A stack is merely a temporary data store, with the limitation that the only data that can be retrieved is the last item to have been stored, i.e. it is a last in first out stack (LIFO). It can be thought of as a tube into which balls can be placed so that the only ball which can be removed is the last one to have been dropped into it. See the sketch.



LAST IN/FIRST OUT STORE
[LIFO]

When the BASIC encounters a GOSUB command it keeps a record of its current program position so that when it reaches the RETURN command it knows where to return to in the program.
It does this by creating a stack of numbers in memory, and when it goes to a subroutine the current program position is stored to the top of this stack. When the RETURN is encountered it removes the top numbers and uses them to find where to return to in the program.

It is possible, of course, to simply use a couple of permanent memory locations to store this number, but what happens when a GOSUB is encountered inside another subroutine? The BASIC works fine when it encounters the first RETURN, but when the second RETURN is found the memory locations no longer have a valid return position. The answer is to have the stack system, whereby numbers are placed onto, and removed from, in the correct order.

```
5 DIM A$(20)
10 GOSUB 1000
20 PRINT "Returned from subroutine."
30 GOTO 10
100 PRINT "This program has been POP
'ed"
200 STOP
1000 INPUT A$
1100 IF A$="EXIT" THEN POP :GOTO 100
1200 PRINT "Still here."
1300 RETURN
```

Let us suppose that your program has entered a subroutine via a GOSUB command, but has decided that it does not wish to RETURN to the point from which it came. If you simply have a GOTO command to exit from the subroutine, this will work but only if the original GOSUB was not itself part of another previous subroutine. If it was, then the BASIC cartridge would really be confused!! The solution is to use the POP command, as this will cause the BASIC to remove the last entry on the return stack, enabling your program to continue out of the subroutine.

RESTORE

The RESTORE function allows the user to use any DATA in the program that may or may not have been READ.
The BASIC keeps track of the DATA to READ next with a pointer. The RESTORE statement simply moves that pointer to the first DATA line in the program.
It is also possible to RESTORE the pointer to a specific line number, i.e. all the DATA on and after that line can once again be READ. It is often very useful to use a variable in place of the line number, as in the GOTO and GOSUB commands, e.g.:

```
10 X=110
20 RESTORE X
30 READ A:READ B
40 PRINT A,B
50 STOP
100 DATA 10,90
105 REM RESTORES TO LINE 110
110 DATA 30,20
120 DATA 100,56
```

TRAP

This command is used to inhibit any error messages which may occur whilst your program is running and allow a continuation of the program at the specified line number (can be a variable as in GOTO X etc.). This function is enabled when the program encounters the TRAP command and can be switched off by using a line number between 32767 and 65535 e.g. TRAP 40000. These errors need not be due to bad programming, however, errors may occur such as trying to plot off the edge of a graphics screen or feeding out data to a printer which does not exist.

```
10 TRAP 1000
20 INPUT NUMBER
30 GOTO 20
1000 REM PROGRAM COMES HERE ON AN ER
ROR
1010 ERR=PEEK(195):REM GET ERROR NUM
BER
1020 ERRLINE=PEEK(186)+256*PEEK(187)
:REM GET ERROR LINE
1030 PRINT "--> ERROR ";ERR;" AT LI
NE ";ERRLINE
1040 PRINT "NOW RETURNING TO RE-TRY"
1090 REM RESET TRAP AND RETURN
1100 TRAP 1000:GOTO ERRLINE
```

Try the above program and type a letter to cause an error. This should illustrate how to use this command to recover from such user errors.

ENTER

ENTER is a powerful command for I/O operations. Its main use can be considered to be that of loading a program into memory. Any valid input device can be used as the source, and this includes screen editor (E:), keyboard (K:), cassette (C:), diskdrives (Dn: filename. ext), RS232 interfaces (Rn:), etc.

A file is ENTERed in the same manner as if it had been typed at the keyboard, so it must be an ASCII file which would look identical to a listed program (a source listing). This feature can be used to 'clean up' a BASIC program. Whilst a program is being written the BASIC automatically creates what is called a variable name table, this being a table of every single variable name that has been used in the program lines. When a program is SAVEd to cassette or disk, the first part of the program saved is this variable name table and the interesting thing is that it may well contain variables which had once been used but have now been deleted. The result is a waste of memory space and time. The solution is to LIST the program to disk or cassette, type NEW, then ENTER the program back. As the program is entered a new variable name table is created including only the variables mentioned in the program. This can then be SAVEd as normal. If you have a program which you have been working on try this and see how much room you gain.

OPEN & CLOSE

These two commands should be used in conjunction with each other as all open channels must be closed after use. A channel is a pathway of two way data from a particular device i.e. Keyboard, Disk, Screen etc. There are a total of eight channels numbered 0 to 7 which can be used for data transfer, but channel 0 can not be closed or opened by the user as the Screen Editor uses this channel for passing data to and from the screen and keyboard. To open a channel you have to specify whether you want read, write or both and any specific control information for that device, e.g. sideways printing on a hardcopy printer. The exact format for OPENing a channel is as follows: first you specify the channel number (1 to 7); next you specify read (4) write (8) or both (12) (there are two more for disk operation); the third is the device control number and is usually 0 (the only exception we know of is the sideways printing option (83)); and the final one is the device name (and for diskettes, the filename as well). It should be realised that although a channel can not be used for a different purpose until it is closed, all available channels can be used simultaneously. This means that more than one *file* on a diskette can be accessed individually.

To CLOSE a channel and free it for other uses you only have to specify the channel number to be closed.

END

This command simply CLOSES *all* channels already open and turns off any sound generators.

NOTE

This command is not for musical notes and is only of use to those of you with disk drives. NOTE enables you to create structured files. All this command does is return two numbers, the first of which is the current sector number, the second is the byte within that sector. The way in which a disk is structured, is that you have 720 sectors to a diskette and 128 bytes to a sector. Each byte contains a number between 0 and 255 (i.e. same range of contents as a memory location). The sector is the 'course' pointer and the byte is the 'finer' pointer to the information stored on the diskette. By saying 'pointer' it has an analogy to the tape counter on a cassette recorder. In the example shown you can see how the program creates a file of names, and at the same time creates another file of positions of these names.

```
10 REM OPEN FILE FOR NAMES
20 OPEN #1,8,0,"D:NAMES.DAT"
30 REM OPEN FILE FOR POSITION OF NAM
ES WITHIN FILE
40 OPEN #2,8,0,"D:POSITION.DAT"
50 REM GET NAMES FROM KEYBOARD
60 REM AND WRITE OUT TO DISK,
70 REM AND NOTE POSITION ON DISK
80 DIM NAME$(20):NUM=1
90 ? "PLEASE ENTER NAME ";NUM
100 INPUT NAME$:IF NAME$="" THEN 150
110 NOTE #1,SECTOR,BYTE:REM FIND SEC
TOR & BYTE IN FILE 'NAMES.DAT'
120 PUT #2,SECTOR:PUT #2,BYTE:REM SA
VE SECTOR & BYTE INTO ANOTHER FILE
130 PRINT #1;NAME$:REM SAVE THE NAME
 INTO 'NAMES.DAT'
140 NUM=NUM+1:GOTO 90:REM GO BACK
150 REM CLOSE DOWN ALL FILES
160 CLOSE #1:CLOSE #2
```

In this simple form the program doesn't offer any great time saving in finding just a particular name in the list. However, when the size of each entry increases to a significant number of bytes (or characters), then selective retrieval is much faster than loading all the data in that file. This is because each pointer is only two bytes long and therefore it is quicker to read to the last pointer than to the last piece of data.

POINT

This is effectively the opposite of the NOTE command. The POINT command instructs the disk drive to go to a particular sector and byte within that sector, enabling you to load into memory a particular byte, thus retrieving the data previously stored in a file.

```
10 REM OPEN FILE FOR NAMES
20 OPEN #1,4,0,"D:NAMES.DAT"
30 REM OPEN FILE FOR POSITION OF NA
MES
40 OPEN #2,4,0,"D:POSITION.DAT"
50 REM ASK FOR THE nth PERSON
60 DIM NAME$(20)
70 ? "PLEASE ENTER THE NUMBER OF TH
E PERSON"
80 TRAP 80:INPUT NUMBER
90 FOR LOOP=1 TO NUMBER:REM DO A DU
MMY READ ON ALL POINTERS THAT ARE N
OT WANTED
100 TRAP 160:GET #2,SECTOR:GET #2,B
YTE
110 NEXT LOOP
```

```
120 POINT #1,SECTOR,BYTE:REM GO TO
NEW POSITION ON DISK
130 INPUT #1;NAME$:REM AND ENTER TH
E NAME
140 ? "PERSON NUMBER ";NUMBER;" NAM
E IS ";NAME$
150 RUN
160 ? :? :? "YOU DO NOT HAVE THAT M
ANY NAMES.....I AM CLOSING ALL FILE
S AND GOING...BYE!!"
170 CLOSE #1:CLOSE #2
```

It is important to realise that this does not give you *true* random access to any sector on a diskette, but limits you to the sectors allocated for files on that diskette (i.e. only those you have 'open' at that time).

PUT & GET

These can be used to transfer a single byte (0 to 255) to and from any device (not just the disk drive). The most common use is to get a value from the keyboard. To do this you must have first opened a channel for reading from the keyboard and then by using the GET command a single value will be returned representing a key code (i.e. not the same as an ASCII code). For example, this is often of use when inputing data without the need to press RETURN. PUT is the opposite of GET and could be used to output characters to the screen. It is, of course, very useful to use these two commands together to transfer data between two devices e.g. Keyboard to Screen, Cassette to Screen, Cassette to Diskette etc.

XIO

XIO enables you to access any operating system file management subsystems i.e. OPENing a file, PRINTing a string, DRAW line, LOCK disk file, FORMAT diskette etc. As you can see, some of these operations have their own command dedicated to them in BASIC because they are most frequently used. Although the ATARI BASIC REFERENCE MANUAL shows a set of pre-determined operations using the XIO command, this is really an extremely useful addition to BASIC. This command provides the only flexible link between the basic language and a user generated device handler. When the BASIC calls the CENTRAL IO VECTOR (CIOV), the command byte is checked to see if a match can be found in the particular devices vector table. For the usual IO functions, i.e. OPEN, CLOSE, GET, PUT, STATUS, there is no complication, CIO simply jumps through the supplied vector, but if no match is found the last vector, called the SPECIAL vector, is entered. Here the user must have altered or generated a routine to test the value of the command byte and 'manually' jump to the user supplied handler sub-routine.

FRE(0)

This command is used to determine the amount of RAM necessary to RUN a program (so that our librarian doesn't have sleepless nights). To do this the point in the program where the most amount of memory is being used at one time (i.e. a graphics mode) is noted and a line must be added at this point in the program in the form of X=FRE(0). When the program is stopped the value of this variable can then be printed (e.g. ?X) and the amount of free RAM can be calculated by taking the amount of RAM you have free before loading the program, and subtracting the value of the variable. This value is in bytes. If you wish to work out how many 'K' are free, then you must divide this value by 1024. The amount of free memory will, of course, still depend on whether you have a disk operating system loaded (DOS), because DOS takes up a certain amount of memory (depends on whose DOS you happen to have), and this means that if you have a cassette system you will have more free memory than on a disk system. Therefore a program which will run on a cassette system may not work on a disk system of the same memory size (before DOS is loaded).

GRAPHICS

The graphics command sets the screen to a pre-designated format with options of having either a split screen (i.e. with a text window), a full screen or no 'clearing' of the screen. These are specified by adding a number (0, 16 or 32 respectively) to the original GRAPHICS mode number, e.g. for a full graphics seven screen you would type 'GRAPHICS 23' or 'GRAPHICS 7+16'. It should be noted that modes 0 and 9 thru 11 are already full screen displays, thus '+16' is not used with these modes. Modes 9 thru 11 are not mentioned in the BASIC manual because when they produced BASIC they had CTIA as their GRAPHICS chip. Since then an improved version has been fitted to all European machines and most of the recent U.S.A. models. This new chip, called GTIA has further GRAPHIC facilities, including a mode of 16 hues, one mode of 16 colours or a mode giving eight colour registers (twice as many as modes 3 thru 7).

COLOR & SETCOLOR

In modes 3 thru 7 there are four colour registers to use. The contents of each of these registers are set to a default value, and to change these values you can use the SETCOLOR command. Each value in the colour register represents a different colour and hue. When the COLOR command is executed it changes the colour being drawn with, to the colour of the value in that appropriate register. If a colour register is altered then all the points drawn using that register are automatically changed. In all graphics modes except mode 9 there are only eight levels of luminance, i.e. 0,2,4,6,8,10,12,14. However, in graphics mode 9 sixteen luminance levels are available, but only one overall colour is available. In graphics mode 10 you have control of eight colour registers which enable you to have eight colours with eight different intensities. Finally, we have included a short demo program to show the use of the colour command.

```
1 REM 'DELAY' CHANGES THE SPEED
2 DELAY=50
10 GRAPHICS 3+16
20 COLOR 1:POKE 765,1
30 PLOT 30,11:DRAWTO 30,0:DRAWTO 10
,0:POSITION 10,11
40 XIO 18,#6,0,0,"S:"
50 COLOR 2:POKE 765,2
60 PLOT 39,23:DRAWTO 39,12:DRAWTO 2
0,12:POSITION 20,23
70 XIO 18,#6,0,0,"S:"
80 COLOR 3:POKE 765,3
90 PLOT 19,23:DRAWTO 19,12:DRAWTO 0
,12:POSITION 0,23
100 XIO 18,#6,0,0,"S:"
110 REM SET COLOURS
120 FOR X=0 TO 4:SETCOLOR X,0,0:NEX
T X:INT=10
130 FOR COL=0 TO 15
140 FOR A=0 TO 2
150 SETCOLOR A,COL,INT
160 FOR B=0 TO DELAY:NEXT B
170 SETCOLOR A,0,0
180 NEXT A
190 NEXT COL
200 GOTO 130
```

LOCATE

Locate can be used in all graphic and text modes to return the value of a specific x-y location on the screen. In text modes it will give the value of the character at those coordinates, and in graphics modes will return the number of the colour register which has been used to draw that pixel. This command can be useful for detecting collisions between objects printed to the screen.

POSITION

This command merely moves the cursor in any graphic or text mode to a specific point on the screen immediately.

SOUND

This command is explained reasonably well in the manual, but there are a few details not mentioned, such as the ability to change the reference frequency from 64KHz to 15KHz. The effect of this is to shift all the sound generator output frequencies down to a lower note. This is achieved by

BYTEING DEEPER INTO BASIC

POKEing the audio control register (53768) with 1. Conversely, a shift in the other direction can be made by POKEing this register with 96, when the reference clock will now be 1.79MHz.

We finish this section with some brief examples of how to generate enveloped sounds.

```
10 REM *** SOUND EFFECT 1 ***
20 FOR V=15 TO 0 STEP -0.07
30 SOUND 0,10*V,10,V:SOUND 1,10*V+1
,10,V
40 NEXT V
50 FOR T=0 TO 100:NEXT T
70 FOR V=0 TO 15 STEP 0.07
80 SOUND 0,10*V,10,V:SOUND 1,10*V+1
,10,V
90 NEXT V
100 FOR T=0 TO 100:NEXT T
110 SOUND 0,0,0,0:SOUND 1,0,0,0
120 FOR N=0 TO 10
130 FOR V=0 TO 15 STEP 0.5
140 SOUND 0,20,10,V
150 NEXT V:NEXT N
160 GOTO 20
```

```
10 REM *** SOUND EFFECT 2 ***
20 AUDCTL=53768:AUDF1=53760:AUDF2=5
3762:AUDF3=53764:AUDF4=53766
30 AUDC1=53761:AUDC2=53763:AUDC3=53
765:AUDC4=53767
35 REM *** SET UP FREQ & LOW TONE *
**
40 POKE AUDF1,255:POKE AUDF2,254:PO
KE AUDF3,253:POKE AUDF4,252:POKE AU
DCTL,1
50 FOR X=1 TO 10 STEP 0.5
60 FOR V=0 TO 15 STEP X
65 REM *** SET DISTORTION & VOLUME
***
66 REM *** (DISTORTION*16+VOLUME) *
**
70 POKE AUDC1,10*16+V:POKE AUDC2,10
*16+V:POKE AUDC3,10*16+V:POKE AUDC4
,10*16+V
80 NEXT V
90 FOR V=15 TO 0 STEP -X
100 POKE AUDC1,10*16+V:POKE AUDC2,1
0*16+V:POKE AUDC3,10*16+V:POKE AUDC
4,10*16+V
110 NEXT V:NEXT X
120 GOTO 50
```

We hope that this article has answered some of the many questions received and that it will assist you in writing your programs.

SKYBLITZ COMPETITION RESULT

Runs in 16K Cassette or 32K Disk

We are proud to present the winning entry in our SKYBLITZ competition, the object of which, was to rewrite the program listed in the last issue. We had an overwhelming response from members and it took many weeks for us to wade through them, but finally the decision was made as to the winner, and we also decided to include the second and third best in the library section so that hard work won't go unrecognised. The winning entry was submitted by Mr. JON BEFF of Manchester, who's superb program impressed us very much. His rewritten SKYBLITZ includes a special Display List, a DLI for background graduated colours, modified character set, and a Vertical Blank Interrupt routine. All these improvements give an overall enhancement to the original game concept. Congratulations to JON, we hope you are getting full enjoyment from your chosen prize.

Our many thanks to all those who submitted programs, we hope you all had many hours of fun rewriting SKYBLITZ and maybe next time you will be the winner!

P.S. Our machine code programmer, Keith Mayhew, would like to thank Mr. D. West for submitting the only 100% machine code version, which gave him great pleasure to judge; he appreciates the hard work that went into producing the program which resulted in a faster and more challenging game. If the visual impact of this version had also been improved, it may well have been the winner.

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

```
10 REM SKYBLITZ  -by Jon Beff
20 REM
30 GOSUB 3000:REM SET-UP
40 GOTO 2500:REM START
100 REM MAIN ROUTINE
110 FOR I=1 TO 4:SOUND 0,114,10,6:SOUND 0,114,10,4:FOR J
=1 TO 180:NEXT J:SOUND 0,136,10,6:SOUND 1,68,10,1
111 FOR J=1 TO 180:NEXT J:NEXT I
112 SOUND 0,0,0,0:SOUND 1,0,0,0
118 POKE 205,16
120 POKE 764,255:POKE 203,1:REM MOVFL
130 IF PEEK(203)<>0 THEN 170
140 POKE 207,0:D=USR(ADR(CL$),PMB+3)
145 FOR I=0 TO 3:SOUND I,0,0,0:NEXT I
150 IF PEEK(206)<>204 THEN 1000
160 GOSUB 2600:GOTO 100:REM NEW SCRN
170 IF PEEK(764)=255 THEN 130
180 POKE 764,255:POKE 207,1:REM BOMB
190 IF PEEK(203)=0 THEN 130
192 A=PEEK(207)+24
200 IF PEEK(207)>0 THEN SOUND 0,A,10,3:SOUND 1,A+1,10,1:
GOTO 190
205 POKE 53761,0:POKE 53763,0
210 REM HIT CITY?
220 IF PEEK(1536)<>0 THEN GOSUB 900
230 GOTO 130
900 REM BUILDING HIT
901 POKE 1536,0:REM CLEAR EXPFL
902 SCORE=SCORE+10*LEVEL:POSITION 7,0:? SCORE
903 X=2*INT(INT((PEEK(1537)-39)/4)/2)
```

SYKBLITZ

```
904 Y=INT((PEEK(1539)-12)/16-1)
906 A=LEVEL*8:REM DELAY
910 FOR I=35 TO 41 STEP 2
915 IF PEEK(203)=0 THEN POP :GOTO 130:REM CRASHED/LANDED
920 POSITION X,Y:? CHR$(I);CHR$(I+1)
923 SOUND 0,I+45,0,48-I
930 FOR J=1 TO A:NEXT J
940 NEXT I
945 POKE 53761,0
950 IF Y=10 THEN POKE 87,7:COLOR 0:PLOT 4*X,11:DRAWTO 4*
X+6,11:PLOT 0,11:POKE 87,0
970 RETURN
1000 REM SPACESHIP CRASH
1020 L=PMBASE+1024+PEEK(206)
1030 POKE 53249,0:REM HPOSP1
1035 POKE 704,14
1050 FOR I=1 TO 2:FOR J=1 TO 6
1051 SOUND 0,60,0,13-J*(I=2)
1052 SOUND 1,120,0,13-J*(I=2)
1056 POKE 704,14-PEEK(704)
1057 POKE 712,10-PEEK(712)
1058 POKE 1791,10-PEEK(1791)
1060 POKE L-4+J,EXPL(J,1)*INT(EXPL(J,2)*RND(0)):POKE L-3
-J,EXPL(J,1)*INT(EXPL(J,2)*RND(0))
1070 NEXT J:NEXT I
1090 FOR I=6 TO 1 STEP -1:FOR J=1 TO I
1091 SOUND 0,60,4,I:SOUND 1,120,4,I
1096 POKE 704,14-PEEK(704)
1100 POKE L-4+J,EXPL(J,1)*INT(EXPL(J,2)*RND(0)):POKE L-3
-J,EXPL(J,1)*INT(EXPL(J,2)*RND(0)):NEXT J
1110 POKE L-4+I,0:POKE L-3-I,0:NEXT I
1111 FOR I=1 TO 40:NEXT I
1112 POKE 53761,0:POKE 53763,0
1113 POKE 53278,0:REM HITCLR
1115 POKE 53248,0:REM HPOSP0
1130 D=USR(ADR(CL$),PMB+5):REM CLR P1
1131 FOR I=1 TO 120:NEXT I
1132 IF SCORE>HIGH THEN HIGH=SCORE
1136 POSITION 13,0:? "HI-";HIGH;"  "
1137 IF SCORE<>HIGH OR SCORE=0 THEN FOR I=1 TO 200:NEXT
I:GOTO 1145
1138 FOR I=200 TO 81 STEP -2:SOUND 0,I,10,8:SOUND 1,I+1,
10,8:NEXT I:FOR J=1 TO 160:NEXT J
1139 POKE 53761,0:POKE 53763,0
1140 FOR A=1 TO 4:SETCOLOR 0,0,0:FOR J=1 TO 60:NEXT J:SE
TCOLOR 0,0,12:FOR J=1 TO 60:NEXT J:NEXT A
1145 FOR I=1 TO 200:NEXT I:GOTO 2500
2000 REM DRAW BUILDINGS
2020 ? CHR$(125);:POKE 559,0
2025 RESTORE 3659+(LEVEL-INT((LEVEL-1)/3)*3)
2026 FOR I=1779 TO 1791
2027 READ A:POKE I,A:NEXT I
2032 SETCOLOR 0,0,12:REM SCORE COLOR
2034 SETCOLOR 1,2,0:REM BRICK COLOR
2036 SETCOLOR 2,0,12:REM WINDOW COLOR
2038 SETCOLOR 3,2,14:REM INTRO COLOR
2040 POKE 82,0:POKE 752,1
2060 POKE 87,7:COLOR 2:FOR I=16 TO 142 STEP 8:PLOT I,11:
DRAWTO I+6,11:NEXT I:POKE 87,0
2070 FOR I=4 TO 34 STEP 2
2080 Y=8-4*RND(0)+2*(I<8 OR I>30)
2090 COLOR 33:PLOT I,10:DRAWTO I,Y
2100 COLOR 34:PLOT I+1,10:DRAWTO I+1,Y
2110 NEXT I
2115 COLOR 32:PLOT 0,1
2120 POKE 559,DMA
2130 RETURN
2500 REM START
2505 LEVEL=0:SCORE=0
2515 GOSUB 2600:I=2590
2520 POSITION 0,0:? "       sKy-bLiTz     "
2530 GOSUB I
2540 ? "-by jon beff"
2550 GOSUB I
2560 ? "PRESS  START"
2570 GOSUB I
2580 GOTO 2520
2590 FOR J=1 TO 150:IF PEEK(53279)=6 THEN POP :GOSUB 280
0:GOTO 100
2595 NEXT J:POSITION 4,0:RETURN
2600 REM NEW SCREEN
2601 IF LEVEL=0 THEN 2620
2602 RESTORE 2605
2603 FOR I=1 TO 7:READ A:FOR J=1 TO 10:SOUND 0,A,10,10:S
OUND 1,A+1,10,2:NEXT J:NEXT I:FOR I=1 TO 30:NEXT I
2604 POKE 53761,0:POKE 53763,0
2605 DATA 243,193,217,182,193,162,121
2606 FOR A=1 TO 60:NEXT A
2607 FOR I=1 TO 10:SCORE=SCORE+10*LEVEL:POSITION 7,0:? S
CORE:FOR J=12 TO 0 STEP -3:SOUND 0,60,10,J:NEXT J
2608 FOR J=1 TO 3:NEXT J:NEXT I
2610 FOR I=1 TO 100:NEXT I
2612 FOR I=1791 TO 1779 STEP -1:POKE I,0
2614 FOR J=1 TO 4:NEXT J:NEXT I
2616 POKE 708,0
2620 LEVEL=LEVEL+1
2625 POKE 704,26:REM P0 COL
2630 IF LEVEL>9 THEN LEVEL=9
2640 GOSUB 16000:REM LOAD PLAYERS
2650 GOSUB 2000:IF LEVEL<>1 THEN GOSUB 2800
2655 POKE 205,0
2660 POKE 206,YOFFS1+9:REM SET PMLVL
2670 RETURN
2800 REM SCORING
2810 POSITION 1,0:? #6;"SCORE-";SCORE;"      ":POSITION 1
3,0:? "ZONE-";LEVEL:RETURN
3000 REM MAIN SET-UP ROUTINE
3005 REM MODIFY DISPLAY LIST
3010 GRAPHICS 0
3020 POKE 559,0
3030 DL=PEEK(560)+256*PEEK(561)
3040 POKE DL+3,6+64
3045 POKE DL+6,6
3050 FOR I=1 TO 10
3060 POKE DL+6+I,5
3070 NEXT I
3080 FOR I=11 TO 18
3090 POKE DL+6+I,13
3100 NEXT I
3120 POKE DL+25,65:POKE DL+26,PEEK(560):POKE DL+27,PEEK(
561)
3500 REM IMPLEMENT DLI
```

SKYBLITZ

```
3520 FOR I=6 TO 15
3530 POKE DL+I,PEEK(DL+I)+128
3540 NEXT I
3542 POKE DL+17,PEEK(DL+17)+128
3544 POKE DL+18,PEEK(DL+18)+128
3546 POKE DL+23,PEEK(DL+23)+128
3550 RESTORE 3580
3560 FOR I=1739 TO 1778
3570 READ A:POKE I,A:NEXT I
3580 DATA 72,138,72,152,72,238,242,6
3590 DATA 175,242,6,189,242,6
3600 DATA 69,79,37,78
3610 DATA 141,10,212,141,26,208,224
3620 REM  NO. OF COLORS
3630 DATA 13,208,5,169,0,141,242,6
3640 DATA 104,168,104,170,104,64,0
3650 REM COLOUR DATA
3660 DATA 48,18,34,50,36,38,40,42,44,46,12,20,0
3661 DATA 48,18,34,50,52,54,56,58,60,62,12,20,0
3662 DATA 112,130,114,132,116,118,120,122,124,126,12,20,0
3680 POKE 512,203:POKE 513,6
4000 REM CHANGE CHARACTER SET
4010 CHBASE=(PEEK(106)-16)*256
4012 REM SETUP M/C CHARSET COPIER
4015 DIM CHAR$(34):RESTORE 4025
4020 FOR I=1 TO 34:READ A
4022 CHAR$(I,I)=CHR$(A):NEXT I
4025 DATA 104,104,133,215,104,133,214,169,224,133,213,16
9,0,133,212,162,4,160,0,177,212,145,214,200,208,249,230
4030 DATA 213,230,215,202,208,242,96
4035 D=USR(ADR(CHAR$),CHBASE)
4038 REM LOAD NEW CHARACTERS
4040 FOR I=0 TO 79
4050 READ A:POKE CHBASE+I+8,A:NEXT I
4060 POKE 756,PEEK(106)-16
4080 DATA 170,190,170,190,170,190,170,190
4090 DATA 168,248,168,248,168,248,168,248
4100 DATA 0,0,170,190,170,190,170,190
4110 DATA 0,0,168,248,168,248,168,248
4120 DATA 0,0,0,0,170,190,170,190
4130 DATA 0,0,0,0,168,248,168,248
4140 DATA 0,0,0,0,0,0,170,190
4150 DATA 0,0,0,0,0,0,168,248
4160 DATA 0,0,0,0,0,0,0,0
4170 DATA 0,0,0,0,0,0,0,0
5000 REM SET UP M/C MEMORY CLEAR
5003 DIM CL$(17):RESTORE 5004:FOR I=1 TO 17:READ A:CL$(I
,I)=CHR$(A):NEXT I
5004 DATA 104,104,104,133,213,169,0
5005 DATA 133,212,160,0,145,212,136
5006 DATA 208,251,96
5010 REM SET UP PMG
5035 POKE 706,12
5040 PMB=PEEK(106)-16
5050 POKE 54279,PMB
5060 PMBASE=PMB*256
5070 POKE 53277,3
5080 YOFFS1=51:YOFFS2=53
5170 DMA=62
5180 POKE 53256,1:POKE 53257,1
6000 REM SET UP EXPL MATRIX
6020 DIM EXPL(6,2)
6030 RESTORE 6080
6040 FOR I=1 TO 6
6050 READ X,Y
6060 EXPL(I,1)=X:EXPL(I,2)=Y
6070 NEXT I
6080 DATA 1,256,1,256,2,64
6090 DATA 2,64,4,16,8,4
10000 REM SET-UP VBI ROUTINE
10010 RESTORE 10040
10020 FOR I=1539 TO 1737
10030 READ A:POKE I,A:NEXT I
10040 DATA 104,160,13,162,6,169,7,76
10050 DATA 92,228,216,169,0,197,203,208
10060 DATA 3,76,199,6,164,207,240,74
10070 DATA 192,1,208,15,165,205,141,1
10080 DATA 6,24,105,8,141,6,208,165
10090 DATA 206,133,208,169,0,145,208,169
10100 DATA 48,162,6,200,145,208,202,208
10110 DATA 250,230,207,24,165,207,101,208
10120 DATA 141,2,6,201,205,240,13,173
10130 DATA 2,208,201,0,240,20,141,30
10140 DATA 208,141,0,6,169,0,164,207
10150 DATA 162,6,145,208,200,202,208,250
10160 DATA 133,207,230,204,169,2,197,204
10170 DATA 208,90,169,0,133,204,230,205
10180 DATA 169,224,197,205,208,53,169,16
10190 DATA 133,205,160,8,162,254,189,0
10200 DATA 1,157,1,1,189,0,2,157
10210 DATA 1,2,202,208,241,136,208,236
10220 DATA 173,192,2,201,16,240,3,206
10230 DATA 192,2,24,165,206,105,8,133
10240 DATA 206,201,204,208,6,169,0,133
10250 DATA 203,240,25,169,0,205,4,208
10260 DATA 240,7,141,30,208,133,203,198
10270 DATA 205,165,205,141,0,208,141,1
10280 DATA 208,141,193,2,76,98,228
15000 REM INITIALISE VBI ROUTINE
15010 FOR I=203 TO 209:POKE I,0:NEXT I
15011 POKE 1536,0:POKE 1537,0:POKE 1538,0
15015 POKE 209,PMB+3:REM BOMHI
15020 POKE 1667,PMB+4
15030 POKE 1670,PMB+4
15040 POKE 1673,PMB+5
15050 POKE 1676,PMB+5
15060 D=USR(1539)
15070 POKE 54286,192:REM NMI ENABLE
16000 REM LOAD PLAYER DATA
16010 RESTORE 16060
16020 FOR I=PMB+3 TO PMB+7:D=USR(ADR(CL$),I):NEXT I:REM
CLEAR OUT PM AREA
16030 FOR I=PMBASE+YOFFS1+1024 TO PMBASE+YOFFS1+1035:REA
D A:POKE I,A:NEXT I
16040 FOR I=PMBASE+YOFFS2+1280 TO PMBASE+YOFFS2+1286:REA
D A:POKE I,A:NEXT I
16050 REM PLAYER 0
16060 DATA 24,60,102,195,0,255,255,0
16070 DATA 195,126,60,36
16080 REM PLAYER 1
16090 DATA 24,60,255,0,0,255,60
16100 RETURN
```

INTERFACE

Talked into it. . .

Dear Users' Group,

Many thanks to all concerned for the first two issues of the club magazine. They kept me busy for a few weeks, which isn't bad going, considering I was "TALKED INTO" buying a computer in the first place. Perhaps I should explain . . .

I am now past the age when I was the answer to the ultimate question of life, the universe, and everything. And until the beginning of December I thought a "byte" was Old English for what Dracula did to young virgins' necks! Then the family did a deal with me, whereby if we bought a computer for Xmas, they would sell their "VCS" to chip in towards the cost. Little did they (or I) know, that two and a half months later I would be labelled "TERMINAL JUNKIE" by my family, ranging over me like vultures, waiting for me to leave the keyboard for a second.

Ken Ward, Norwich.

COMMENT

Dear 'TERMINAL JUNKIE',

We know exactly how you feel. We too have had the same illness, but we would like to reassure your family that this phase will pass and eventually you will only be at the keyboard for five or six hours a day!!! At that stage the rest of your family may get a chance to suffer from bruised fingertips!

Good luck to you and yours and keep paying the subscription!

Prestel?

Dear Sirs,

First, many thanks for issue 2 of the UK ATARI OWNERS' NEWSLETTER, it was worth waiting for. A fine improvement on the first issue. I was impressed by the layout of the text and illustrations, especially in the Player Missile Graphics article. Re: Keith Berry's letter concerning the creation of a library of subroutines by extracting lines from an existing program, you can use the command LIST "device", linenum 1, linenum 2 to move particular lines out of a program to cassette or disk. Referring to Janet Kemsley's letter (Gosh!! a female ATARI enthusiast) concerning the use of the audio track on cassette programs, excellent references may be found in the American magazines: "Antic" issue 4 & "Compute" issue 14. "Antic" is devoted to ATARI only and complements "ANALOG". "Compute" is primarily based on all 6502 micros, but has always had a strong emphasis on the developing ATARI.

I have sent you a program for your/our software library, it is a character generator for the "Dungeons & Dragons" fantasy role playing game which I wrote recently. I hope that some of the other group members will find it useful. Finally, can I say that I would be interested in seeing any information on Prestel access via modems etc, using the ATARI machines. I understand that Prestel is setting up a micronet service for home computer enthusiasts to be available in 1983. Perhaps the newsletter could go online!

Once again, keep up the good work, and I look forward to a shorter wait for newsletter 3.

Allen J. Palmer, Basingstoke.

COMMENT

Thanks Allen for your interesting and informative letter. Your program "Dungeons & Dragons" has been added to our library so that other members can feel the benefit of your hard work. We at "HQ" have no information at the moment regarding accessing PRESTEL, perhaps someone out there has some information. If so, send details to us and we will pass it on to Mr Palmer.

P.S. Hope you are even more impressed by this issue.

Beginners' Comment

Dear all,

We have only owned our ATARI 400 for a short time. We have gone through the 'ATARI BASIC' which was with our BASIC cartridge but there are still a lot of things we do not understand about graphics. Could you advise us on the next step to take to help us program our own games e.g. what books or cartridges to buy?

Also the character set redefinition program in your first issue does not do what you say it will on our computer. We modified it as you told us to in issue two, but after we had put in '? PEEK (38920)' the computer prints '152' not '0', and the exclamation mark is just the same after the program as before. Does this mean there is something wrong with our computer? Hope you can help us.

Janet and Andy Few, Birmingham.

COMMENT

The way to learn about graphics is not to copy other people's programs and then try to understand exactly what is happening and why. It is best to invest in a book (or two) that covers all the concepts and useful techniques of graphics together with experimentation of your own. For this use we would recommend 'De Re ATARI'. Although this is an expensive manual, it covers just about everything on graphics and is well worth the money. Other books which may be of interest are 'ATARI SOUND and GRAPHICS', 'YOUR ATARI COMPUTER' and a book devoted to game listings called 'GAMES FOR THE ATARI', all of which are available from Maplin. If you find all these books hard work there are several programs which also attempt to explain the ins and outs of this complex subject such as 'INVITATION TO PROGRAMMING 3', 'BASICS OF ANIMATION', 'PLAYER MISSILE GRAPHICS' and 'TRICKY TUTORIALS', to mention but a few.

Re: character set program in issue one. Using the modification in issue two the program copies the existing character set into a place available on a 16K machine and then tells the machine to use it. Unfortunately gremlins swiped us a mighty blow, and whilst in a stunned condition we overlooked one last thing. In line 30 the program is trying to change a location which does not exist on a 16K or 32K machine, so change it to read:

30 POKE 14344+A, BYTE

This now changes the character set you have just copied (which starts at 14336).

Copying. . .

Dear Sirs,

Whenever I buy or write a program on cassette I always make a copy, but with "Galactic Chase" (machine code), to load this program you remove any cartridge and switch on holding down the start button, so I am unable to list or save it on another tape. If I buy the Assembler Editor cartridge, will I be able to list and save it, or do you know a way I can do it with the BASIC cartridge.

Also, concerning disk drives, are they going to come down in price as the computers have, because the 810 nearly costs more than the computer? Another little query of mine is if a program, e.g. Deadline, has two disks, does this mean I will need two disk drives to play the game? If not why are there two disks?

With reference to issue 2 hardware news, a typewriter keyboard is available from quite a few computer shops now (I think the touch one is much better any way and easier to use). Finally, congratulations on a very useful and helpful newsletter, I patiently await the next issue.

A. Clarke, Stoke-on-Trent.

COMMENT

It is a good idea to make back-up copies of your own programs, but if you buy pre-recorded software you will find that the majority of it is protected from being copied. This is because the manufacturer has spent a large amount of time and money in development and is unhappy to have people making their own copies (even if only for a back-up).

We know that some people are capable of copying such software, but we would like to remind you that this is illegal.

We think that the price of the ATARI 810 disk drive is not likely to drop much (if at all) in the near future. The main reason being is that unlike the computer it is mainly mechanical hardware. At approx. three hundred pounds a disk drive seems like an expensive proposition, but we would like to point out that the 810 does not need any extra interfaces or masses of connecting leads, unlike some other makes. Also on many other machines the Disk Operating System (DOS) is held in ROM, which gives you no ROM to manoeuvre (these are the jokes folks!!!), which means that if you wish to use a better DOS it is expensive and fiddly to change ROM chips. ATARI put DOS as a file on the diskette, which automatically 'boots in' at power up, thus giving the user the capability of either changing for a more advanced DOS or modifying the existing one. Of course, a disk drive also gives you greater speed, reliability, filenames and random access capabilities as opposed to the cheaper cassette system.

P.S. 'Deadline' and other similar programs which use more than one diskette do only use one drive, one disk for the program and one for the data.

Useful Tips

Dear all,

Thank you for an interesting second issue. I have a few points on other members' letters and on your article on PM graphics. Firstly, I have found that "PEEK(106)-16" is not a passport to clear screen when using PM graphics. I have had to use trial & error and multiples of eight to try to get rid of those annoying little lines (it took me a month to work out what they were). I've found "PEEK(106)-24" to be the best with my 16K machine.

For those ATARI owners who spend long lonely nights tucked away in quiet corners of the house, there is another use for "POKE 54018,52". Stick your favourite music cassette into your ATARI cassette deck, press play and listen to your music through the television speaker.

Your machine code routine accessed via BASIC for vertical player movement was much appreciated. I knew there had to be a quicker way!! If lines 230, 240, 260 & 270 are slightly modified diagonal movement is possible. Just increase the data in brackets as follows:

```
LINE 230 -(S=11) becomes -((S=11)
      +(S=10)+(S=9))
LINE 240 +(S=7) becomes +((S=7)
      +(S=6)+(S=5))
LINE 260 -(S=14) becomes -((S=14)
      +(S=10)+(S=6))
LINE 270+(S=13) becomes +((S=13)
      +(S=9)+(S=5))
```

Finally, could I make a request for an article on scrolling, particularly when graphics modes higher than 2 are being used. Games like "Rearguard", written in BASIC, use very effective scrolling techniques, perhaps you could pass on the secrets? Keep up the good work.

Hugh Denholm, Aberdeen.

COMMENT

Re: 'PEEK (106)-16', you are perfectly correct, this is not a guaranteed way of obtaining a clear screen in all modes. Unfortunately when we wrote the article, we found it worked perfectly in GRAPHICS 0 but somehow we forgot(!) to check the other modes. PEEK (106) points to the top of memory, if you minus sixteen 'pages' this is usually enough to give you adequate space in which to use your players. In modes that use a large amount of memory e.g. GRAPHICS 8, if you only subtract 16, this will mean that your players will overlap into the screen memory thus causing spurious lines occurring in your players. Our thanks to you Hugh for pointing out the use of minus 24 for these higher modes, but note this applies to any size machine.

We will be publishing an article concerning scrolling techniques in a future issue so keep on reading. Also if there is any one else who would like to see a subject covered or indeed a more detailed explanation of one we have already published, write to us and we will see what we can do!

Queries?

Dear Sirs,

Could you please answer the following questions:—

(a) Could you identify all of the pins on each socket on the ATARI 800, stating functions and any voltage ranges?

(b) In the case of the joystick ports I understand that some pins are analogue inputs. Is this correct and if so what is the voltage range (or is it resistance range) and what pairs of pins are used for it?

(c) Is any video monitor suitable for use with the ATARI?

(d) Could I input speech data into the joystick A/D port directly from a microphone/amplifier and display the resultant waveform?

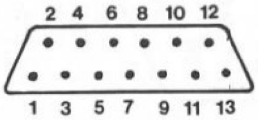
(e) Are there any cables to connect directly from the ATARI to a Centronics printer?

From G. Lawrence — Enfield.

COMMENT

(a) Here is the information for the Serial I/O port and the Monitor Jack, see the article "THE ATARI PORTS UNCORKED" elsewhere in this issue for the details of the joystick port:-

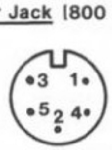
Serial I/O Port



1 Clock In
2 Clock Out
3 Data In to computer
4 Ground
5 Data Out from computer
6 Ground
7 Command
8 Motor Control
9 Proceed
10 +5V/Ready
11 Audio In
12 +12V
13 Interupt

All these pins are 0 to +5Volts except pin 12

Monitor Jack [800 only]



1 Composite Luminance
2 Ground
3 Audio Out
4 Composite Video
5 Composite Chroma

(b) The ATARI can indeed accept analogue inputs, in fact it has eight of them, two per jack. These are pins 5 and 9, and their voltage range is 0V to +5V.

(c) If you wish to connect a video monitor, it must be PAL composite video (pins 2 and 4 on the monitor jack). If you have sound on your monitor/hi-fi then use pins 2 and 3.

(d) Speech data can be sampled and displayed by the ATARI. It is probably best to use an external A/D converter as the internal ones scan at a slower rate than would be required. If you do use the internal A/D converters, these may just about manage the required sampling rate if put into their "FAST SCAN MODE". To monitor and/or display the data you will need to use machine code.

(e) There is a Centronics parallel printer interface cable available which plugs directly into jacks 3 and 4. However, this interface does use a machine code boot program to drive it. This is O.K. but when it is necessary to use other auto-boot software, such as ATARI WORD PROCESSOR, the machine code controller cannot be loaded. Thus this interface acts just like the 850 interface unit.

Phantom Flan Flinger

Dear Chris,

I purchased my ATARI 400 just before Christmas and naturally I joined the "THE UK ATARI COMPUTER OWNERS' CLUB", and I started my subscription from issue 1 of the newsletter. In the first issue I was pleased to see a rather lengthy program entitled "PHANTOM FLAN FLINGER", which I started to type into my machine. After one and a half hours, I typed in RUN, up came the instructions for the game, followed by "When The Saints Go Marching In". Then a catchy tune played, displaying the title of the game, followed by "PRESS START" flashing in inverted and normal lettering. After pressing start a very quick burst of something that can be regarded as ordinary squares on a background of yellow. THEN NOTHING JUST COMPLETE YELLOW!!!

The BREAK key did not make any difference nor did the SYSTEM RESET. The only solution was to switch off the machine and lose the program. I have retyped the program and saved it on my cassette, I have double checked the program and I have found no mistakes, but it still will not run. I would be grateful if you could shed some light on my dilemma.

Another thing I would like to know is that there is a slide switch on the back of the ATARI console which I cannot find labelled in the manuals, what is it for?

Julian Stanley, Worcester.

COMMENT

Sorry to hear you are having problems, we think we know why. Unfortunately some people were sent a copy of issue 1 that had a page missing and guess what!! That's right, it was the last part of the "PHANTOM FLAN FLINGER" listing. If anybody else out there is missing a page (the last line number in the listing should be 14040) write in to us and we will send you a copy.

The switch on the rear of the 400 (on the side of the 800) is for VHF channel group select, since the ATARI for the UK market has an UHF modulator, this switch is not necessary and indeed in the UK machines it is not actually connected to anything. Obviously in the USA there are a vast number of channels available throughout the country, so the user would select his local channel.

CPY again!

Dear All,

I want to start experimenting with simplified input devices which children can use (e.g. keyboards with just "yes", "no" and "help!" keys). I've already dismantled 2 broken joysticks to cannibalise the plugs and cables, but would like a more reliable, and less expensive source. I know that the 9-pin connector plugs and cables are available relatively cheaply in America, but do you know if spare parts are available in the UK?

Following on from your article on modifying character sets in issue 1, here is a nice quick machine language routine to copy the entire set from ROM into RAM, starting at address CHSET.

```
10 FOR A = 1536 TO 1569: READ D:
      POKE A, D: NEXT A
20 DATA 104, 104, 133, 204, 104,
      133, 203, 169, 0, 133, 205, 169,
      224, 133, 206, 162, 4, 160, 0
30 DATA 177, 205, 145, 203, 136,
      208, 249, 230, 204, 230, 206,
      202, 208, 242, 96
40 X = USR (1536, CHSET): POKE
      756, CHSET/256
```

This routine runs in less than one second. To get the modified upper case set in Graphics 1 or 2, POKE 756, CHSET/256 after the graphics command. To get the new lower case set, POKE 756, CHSET/256+2.

Finally, I too like Mr Williams in issue 1 find that my Assembler Editor cartridge will not assemble CPY in immediate mode. The instruction works in other addressing modes, and CPX works perfectly. At the moment I have to introduce more complex routines to get around the problem, particularly when addressing tables from a page zero vector. Any suggestions?

Len Golding, Sheffield.

COMMENT

Spare 9-pin 'D' connectors and multi-core cables have been available for some time in the UK from Maplin Electronics. Good luck with your experiments, if you are successful let us know and we will be pleased to pass on details to others.

With regard to your assembler/editor cartridge problem, this is a bug in the cartridge but the latest may not have this bug. There are now two other assemblers, both on disk, 'SYNASSEMBLER' which is like a faster ASSEMBLER EDITOR, or 'MACRO ASSEMBLER' which is a very powerful development tool and superior to the others in our opinion.

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

THE ATARI PORTS UNCORKED!

On the front of the ATARI there are 4 "CONTROLLER JACKS". Most people assume these are merely for games controllers e.g. joystick & paddle use. The true power of these ports has never really been exploited, but now all will be revealed.

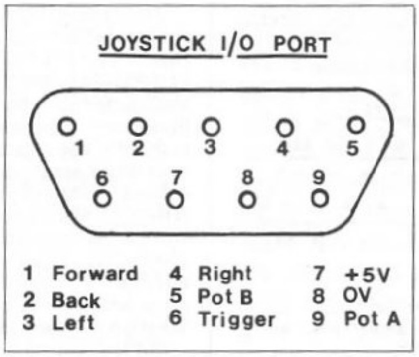
The ATARI has two eight bit parallel input/output ports and each port is split between two controller jacks. This means that port A is jacks 1 & 2 and port B is jacks 3 & 4. These two ports may be set up for input or output or in/out simultaneously. Also, there are two "trigger" inputs per port, i.e. one per jack. There are two analogue to digital converters per jack as well (giving a total of eight). The triggers go into GTIA and the A to D converters go into POKEY.

For ports A & B there are two registers, these are the DATA register and the CONTROL register. There names are PORTA & PORTB (DATA registers), PACTL & PBCTL (CONTROL registers).

ADDRESS LOCATIONS

Name	Hex	Decimal
PORTA	D300	54016
PORTB	D301	54017
PACTL	D302	54018
PBCTL	D303	54019

The following explanation is for PORT A (jacks 1 & 2) but is equally valid for PORT B (jacks 3 & 4).



The control register (PACTL) has only one bit used for ports, as the rest are for use with the serial bus on the side of the computer and will not be covered in this article. The bit relevant to the ports is bit 2 . This is set to one when the computer is initially turned on.
When this bit is set to one the PORT register is used for reading the data at jacks 1 & 2 on the front of the computer.
If bit 2 in the control register is now reset to zero then this alters the function of the PORT register and it is now used for setting a new input/output pattern for jacks 1 & 2. Initially this direction pattern is all zeros and gives input on all eight pins. If you write a byte to the PORT register, each one and zero will represent output and input respectively. Thus individual pins on the same jack can be configured for input or output. Once this pattern is written to the PORT register then bit 2 of PACTL must be set back to a one so that the PORT register is now ready for use as input and output.

So, as you can see, the three basic steps for changing the input/output pattern are:

1. Set PACTL bit 2 to zero.
2. Write required byte (bit pattern) to PORTA.
3. Reset PACTL bit 2 back to one.

To go about setting and resetting bit 2 of PACTL (or PBCTL) you must POKE the appropriate register with 60 to set bit 2 to one and 56 to reset bit 2 to zero, e.g. POKE 54018,56 sets PACTL bit 2 to zero.

ANALOGUE TO DIGITAL CONVERTERS

As mentioned, the ATARI has eight analogue to digital converters. These enable the ATARI to interpret information coming in from the 'real' world. This information is in the form of a varying voltage which is translated by any one of the eight A to D's into an eight bit number.

Paddles are merely potentiometers which alter the input voltage to a particular A to D. The eight A to D's are divided into pairs on the jacks. There is no need for the BASIC programmer to know the hardware locations of these as all the A to D's can be read using the PADDLE command.

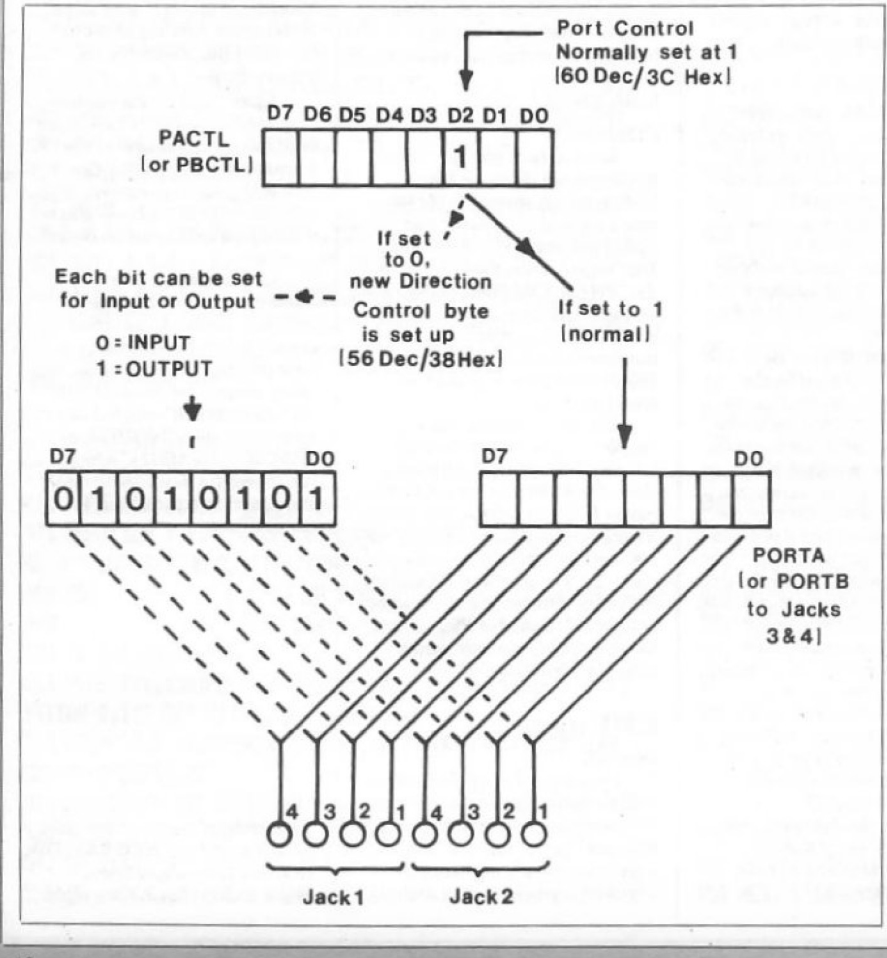
The sampling rate of these are slightly less than 0.02 of a second. If they are to be used with BASIC then this sampling rate is more than sufficient. It is possible to increase this speed for machine code programs to about 3.5 KHz. The reason you may require this faster speed is for such things as speech recording and reproduction etc. An example of using the normal speed from BASIC is demonstrated in the LIE DETECTOR program in the software library.

TRIGGERS

Again this sounds GAMEish! But, the trigger input of all the jacks do, in fact, have two functions. Firstly, they act as conventional triggers on the fire buttons, but also when any of these inputs go low (fire button is pressed) then the current scan line counter is transferred into PENV and the current position along the screen is transferred into PENH (locations 565 and 564 respectively).

Unless you haven't already guessed these are the light pen locations!
This means that the ATARI is capable of using a light pen directly, until recently there has not been one available on the British market, now they are beginning to appear, at around £25.

In a future issue we hope to present the hardware design for your own light pen!



CUSTOM KEYBOARDS

by Len Golding — Sheffield.

Fancy a new keyboard for your Atari? No, I don't mean one of those £80 add-ons for the 400, I mean an extra set of keys *in addition* to the ones on the console, for about a fiver. If you can handle a soldering iron it's easy to build a custom keyboard with up to 31 keys, and only a little harder (and more expensive) to add a great many more.

Fine, but why bother? Well, for a start, custom keyboards with clearly labelled keys are a lot easier for naive users to understand than conventional "press Y for yes or N for no, then press RETURN" instructions. Ever had anyone crash your program by typing "Y" in inverse or lower case, or by accidentally pressing BREAK or SYSTEM RESET? A custom keyboard containing only a handful of keys virtually eliminates the risk of these non-valid entries. They can be used to provide special facilities not readily available from the console — how about colour-coding for small children, or providing single-touch commands for handicapped (or very busy) people? Or do you fancy a 'remote' link which allows the general public to use your machine while the console is safely tucked out of harm's way — think of the possibilities for fund raising!

JOYSTICK PORT

The secret lies, of course, in the joystick ports. You've paid for four of them, so why not get a little extra value. If you look into a port from the front of the console, you will see 9 pins. Each of these can be connected to outside apparatus by plugging in a standard D-type socket with an appropriate length of multi-way cable. Figure 1 shows the pin functions, together with the colours used by Atari for wiring between external apparatus and the console ports. This is the view you'll have while soldering wires into the back of the connecting socket.

Pins 1 to 4 handle the 4 joystick position switches (A, B, C and D), though pins 3 and 4 double-up as trigger inputs when paddles are connected. Pin 6 (orange wire) is the joystick trigger input, while pin 8 (black) is the common return for all five switches.

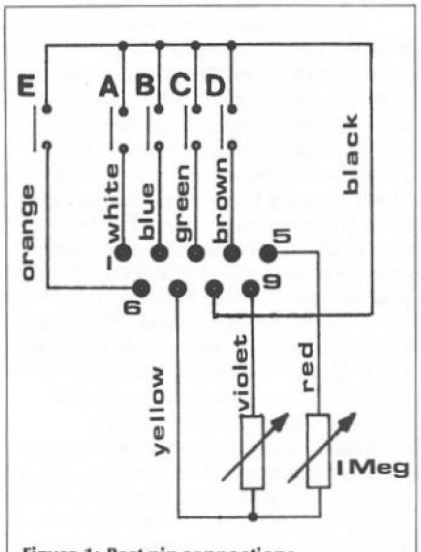


Figure 1: Port pin connections.

Pins 5, 7 and 9 handle the two analogue (paddle) inputs and are included for completeness, though they play no part in the keyboard construction.

Look first at switches A to D. These control four independent data lines which combine to produce a 4-bit binary 'word' with 16 possible variations (including the "all off" state). Atari's joysticks work by mechanically pressing the right combination of switches for any given stick position. However, for keyboard inputs we can only use one key at a time, so we need a hardware device which will convert a single 'on/off' key press into its appropriate 4-bit word.

DIODE MATRIX

Now there are lots of complicated and expensive ways of doing this, but I have found in practice that a very ordinary diode matrix is perfectly satisfactory and, with suitable diodes at about 4p each, the cost is certainly not excessive.

Figure 2 shows the wiring for 12 extra keys, but the pattern should be clear enough to extrapolate for up to 15 keys if you require them. In building the 4-bit word, line A is worth 1, line B is worth 2, line C is worth 4 and line D is worth 8 — an ordinary binary progression in fact.

I use 1N4148 diodes because they are cheap and designed for this type of application, but most small-signal silicon diodes should work. Just make sure they are wired the right way round — i.e. with cathodes to the switches. If in doubt, leave wiring the diodes until last, then try one both ways round to see which works.

Any decent push-to-make switch will do, but you can avoid a lot of tedious interwiring by using the kind that plugs directly into Veroboard, e.g. Maplin's "click switch" FF87U.

JUST LIKE A JOYSTICK

To the computer, a new keyboard of this kind looks just like a joystick with more (or less) positions. Consequently the software to handle it is extremely simple. If you use the left-hand port, a number from 0 to 15 will be returned to address 632. This number will be 15 if no keys are pressed; pressing key 1 makes it 14, key 2 makes it 13 and so on. Of course, there will always be someone who tries pressing more than one key at once and a simple diode matrix can't handle two-key roll-over. If this is likely to prove a serious problem, you could always replace the diode matrix with a keyboard encoder chip (e.g. MM74C922), though this would require an external power supply. Alternatively, a certain amount of software protection can be provided to detect unacceptable inputs. Accessing the keyboard output from BASIC will do for most purposes but, if you want really fast action, you could insert a machine code routine into the vertical blanking period. In this way you can scan the keyboard 50 times per second, while your BASIC program is doing something entirely different.

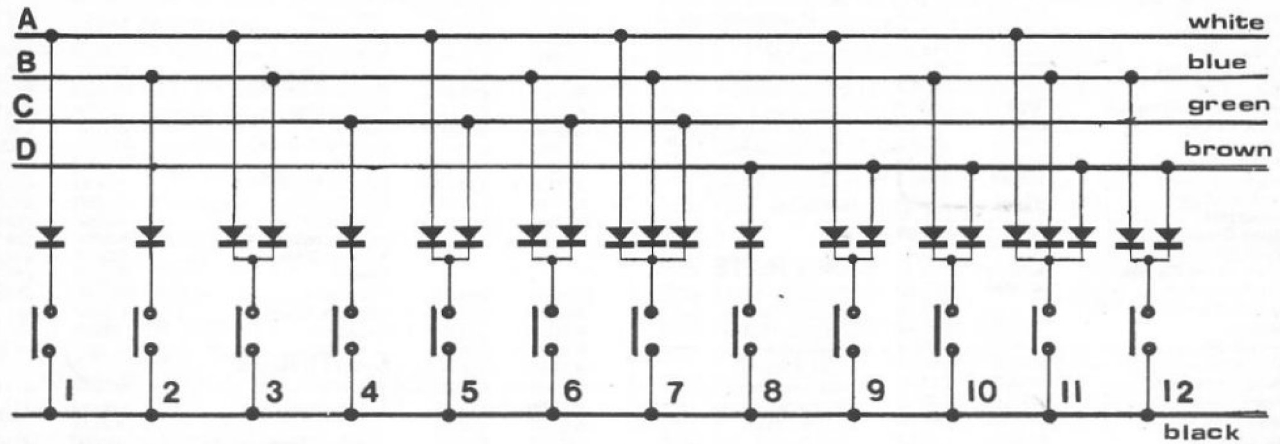


Figure 2: Diode Matrix.

LOTSA KEYS

How about more than 15 keys? (A synthesiser keyboard for example.) There are two options open; you can either tie in the joystick trigger line (orange wire; switch E in Figure 1) or use more than one input port. Using the trigger line in addition to the 4 'position' lines allows a 5-bit word to be read through a single port — this can handle up to 31 new keys. The associated software must give the trigger input a value of 16 (continuing the 1, 2, 4, 8 sequence). This is easy to do, since the trigger address (644 for STRIG Ø) can contain only a 1 or a Ø. Just multiply the contents of the STRIG address by 16, and add the result to the contents of its associated STICK address — e.g.:

```
    N = STICK (Ø) + STRIG (Ø) * 16
or  N = PEEK (632) + PEEK (644) * 16
```

If you need more than 31 keys (e.g. for a 4-octave keyboard), simply use two joystick ports and write software to combine the 'position' lines into an 8-bit word — e.g.:

```
    N = STICK (Ø) + STICK (1) * 16
```

CUSTOM KEYBOARDS

You'll need another of those D-type sockets for this, of course.

CONNECTING UP

Which brings me to the only tricky bit in the whole procedure — getting the connectors to work. The problem comes when you fit the socket into its protective shell. Two self-tapping screws are used for this purpose, and the heads of these screws butt up against the plastic shield which Atari use to pretty-up the joystick ports. If you remove this shield there is no problem (except perhaps for your guarantee), and standard sockets will push in far enough to make a reliable connection. Alternatively the offending screw heads can be ground down, or the socket glued into its shell. Atari will supply a ready-wired socket and cable assembly, though it's as cheap to cannibalise old joysticks. My solution is to solder short lengths of 4BA brass studding (Maplin can supply it) into the fixing holes so that they are flush with the front surface of the fixing plate, and extend backwards about 75mm. The wired assembly can then be bolted into a home-made shell from the back. Suitable shells can be made from perspex sheet or rectangular aluminium tube. For the wiring, use ribbon cable or a bunch of single wires; multi-way screened cable may look attractive but it is very fiddly to solder and too flimsy to cope with the mechanical strains of this application.

FINALLY. . .

This article has dealt only with keyboards, but there are many other uses for your joystick ports. For example, the mechanical keyboard switches could be replaced by electronic ones, or by relays, allowing you to monitor environmental changes (heat, light, sound, humidity etc.). The paddle inputs can also be used for this purpose — experiment and see. Also, your Atari can use the ports as *output* sockets — switching the data lines on and off under software control (think of the applications in robotics!). That, however, is another story.

SPECIAL OFFERS

Great news for members!! As many of you are aware, although the club is independent from MAPLIN, we enjoy a great deal of moral and technical support from the MAPLIN management. Indeed, two of the top executives are members of the club and own ATARI's. Thanks to their policy of promoting the ATARI computers and this club in particular, they have made available to us software at amazingly low prices, for you to purchase on special offer.

To take advantage of this offer, write the code numbers of the titles you require, together with quantities and prices, onto a piece of paper or use the coupon provided on the insert; and send your order to: CHRIS BARLOW,
THE U.K. ATARI COMPUTER OWNERS CLUB, P.O. BOX 3,
RAYLEIGH, ESSEX.

Make cheques payable to M.E.S. Ltd.

DISKETTE

Ali Baba 40 Thieves	BQ78K	£17.00
Andromeda	KB17T	£13.00
Anti-Sub Patrol	KF67X	£16.00
Armour Assault	KB99H	£17.00
B1 Nuke Bomber	BG69A	£11.00
Baja Bussies	KB75S	£14.00
Basic Routines	KB85G	£11.00
Battle Normandy	KB39N	£19.00
Battle Shiloh	BQ97F	£16.00
Bug Attack	BG37S	£14.00
Bug Off	KB62S	£13.00
Canyon Climber	BG45Y	£16.00
Clowns & Ballons	KB80B	£16.00
Crossfire	BG23A	£13.00
Crush Crumble	BQ84F	£13.00
Crypt Of Undead	KB33L	£13.00
Curse Of Ra	BQ90X	£08.00
Cypher Bowl	BQ20W	£19.00
Cytron Masters	KB42V	£19.00
Datestones Ryn	BQ82D	£08.00
Deluxe Invaders	BG34M	£17.00
Disk Manager	BG58N	£15.00
Danger In Drindisti	KF05F	£08.00
Dragon's Eye	KB32K	£13.00
Escape From Vulcan Isle	KB30H	£13.00
Galactic Chase	BQ61R	£11.00
Genetic Drift	KB66W	£15.00
Gomoku	BG55K	£11.00
Goodcode Cavern	KB04E	£13.00
Guns Fort Dfnc	KB96E	£12.00
Haunted Hill	BG39N	£11.00
Invasion Orion	BQ81C	£10.00
Jawbreaker	BQ26D	£13.00
Journey to the Planets	KB29G	£13.00
Keys Of Acheron	KF07H	£08.00
King Arthur's Heir	KB34M	£13.00
Labyrinth	KB72P	£15.00
Lunar Lander	BG49D	£10.00
Matchracer	BG32K	£14.00
Midway Campaign	BG82D	£11.00
Morloc's Tower	KF15R	£08.00
Mouskattack	BQ77J	£14.00
Music Lessons	KF43W	£12.00
NA Convoy Raider	BG84F	£11.00
Nukewar	BG88V	£11.00
Pacific Highway	BG14Q	£16.00
Paint	KB22Y	£20.00
Pathfinder	BG33L	£16.00
Poker Solitaire	BG53H	£08.00
Preppie	KB08J	£14.00
QS Forth	YL29G	£39.00
Race in Space	BG20W	£10.00
Rasterblaster	BG35Q	£12.00
Rear Guard	KB48C	£11.00
Rescue Rigel	BQ80B	£13.00
Reversi	BG54J	£11.00
Ricochet	BG48C	£08.00
Sea Dragon	KB59P	£15.00
Sea Fox	KB57M	£19.00
Shattered Alliance	BQ98G	£16.00
Shoot Arcade	BG16S	£16.00
Softporn Adventure	BQ93B	£14.00
Space Chase	BG43W	£07.00
Star Warrior	BQ79L	£17.00
Stellar Shuttle	KB46A	£18.00
Stratos	KB54J	£15.00
The Cosmic Balance	KF09K	£19.00
The Nightmare	KB35Q	£13.00
Threshold	BG18U	£18.00
Tigers in Snow	KB02C	£19.00
Tumble Bugs	BG46A	£16.00
Tutti Frutti	KB64U	£11.00
Ulysses Gold Fleece	BQ92A	£14.00
Upper Reaches	BQ88V	£08.00
Wizard and Princess	BQ25C	£15.00
Wordrace Access	KB10L	£10.00
Wordrace	KB09K	£13.00
Zork II	BQ95D	£18.00

CASSETTE

Acquire	BG66W	£09.00
Angle Worms	BG50E	£07.00
Anti-Sub Patrol	KF66W	£11.00
Apple Panic	KB92A	£15.00
Baja Buggies	KB74R	£14.00
Basic Routines	KB84F	£11.00
Battle Normandy	KB38R	£19.00
Battle Shiloh	BG63T	£16.00
Bug Attack	BG36P	£14.00
Bug Off	KB61R	£13.00
Chicken	BG27E	£14.00
Clowns & Ballons	KB79L	£16.00
Conflict·2500	BG85G	£08.00
Crush Crumble +16K	KK10L	£13.00
Curse of Ra	BQ89W	£08.00
Cypher Bowl	BQ20W	£19.00
Cytron Masters	KB41U	£19.00
Datestones Ryn	BQ22Y	£08.00
Danger in Drindisti	KF04E	£08.00
Fast Gammon	YL33L	£10.00
Galactic Chase	BQ62S	£09.00
Genetic Drift	KB65V	£15.00
Ghost Hunter	BQ64U	£11.00
Golf Challenge	KB82D	£12.00
Gomoko	BQ18U	£08.00
Guns Fort Defiance	BG78K	£09.00
Haunted Hill	BG38R	£10.00
Invasion Orion	BQ23A	£10.00
Journey to the Planets	KB28F	£13.00
Keys Of Acheron	KF06G	£08.00
King Arthur's Heir	KB97F	£13.00
Labyrinth	KB71N	£15.00
Matchracer	BG31J	£14.00
Morloc's Tower	KF14Q	£08.00
Music Lessons	KF42V	£12.00
Poker Solitaire	BQ17T	£06.00
Program Aids Pack	BG60Q	£07.00
Rescue Rigel	BQ21X	£13.00
Reversi	BQ19V	£08.00
Ricochet (+8K)	KK15R	£08.00
Ricochet	BG47B	£08.00
Rescue Rigel +16K	KK08J	£13.00
Shoot Arcade	BG15R	£16.00
Shoot Gallery	BQ36P	£09.00
Stellar Shuttle	KB45Y	£18.00
Strange Odyssey	BQ05F	£12.00
Stratos	KB53H	£15.00
The Count	BQ04E	£12.00
Treasure Quest	KB36P	£07.00
Voyager 1	BG95D	£09.00

CARTRIDGE

Alien Garden	KF00A	£17.00
Deluxe Invaders	KB89W	£21.00
Gorf	KB44X	£23.00
Monster Maze	KF01B	£17.00
Plattermania	KF12N	£17.00
Wizard Of Wor	KB94C	£23.00

USER GROUP SOFTWARE

Software Librarian Roy Smith

Welcome to the software/listings section of the newsletter. As you can see this section has changed hands and is being chaperoned by ROY, i.e. me, the main reasons for this are that (1) RON LEVY'S expertise at article writing and programming were not being used to the full and (2) I was bribed to take over (the thought of all that free software was too much). I took over just after Xmas and have worked like a beaver to clear the backlog of people waiting for return programs. The list of programs available to members has increased this quarter and we are presenting them in a different format from now on, we will be giving a bit more information on each title so that you have some idea of what to expect. Of course it must be kept in mind that all of these programs have been written by other members and cover a wide spectrum of capabilities, technique, and subject matter. Actually many of the programs are very good quality and easily compare well with programs published in the glossy magazines!! Indeed, a couple have been, congratulations to HUGH DENHOLM (SUBMARINE HUNTER) & KEITH BERRY (PLAYER DESIGNER). We hope this encourages others to try their luck as we feel the ATARI is underexposed in the national magazines.

Now lets get to the nitty gritty, the basic rules of the library are as follows:

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

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

A few points on the programs themselves: I like to use lines 0 to 9 for administrative purposes so start your program from line 10. If you have more than 16K, try to ensure that any player/ missile graphics or character set tables are placed in as low a place in memory as possible. Work out how much RAM is required to run the program (see BASIC article for details) and let me know. Another thing to bear in mind is that you know how to use your program but other members who request a copy may not, so if any precise instructions are required for your program try to insert this information in the listing using REM statements.

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

CONTACT

We have had many requests from members for more contact with others in their areas. We think this is a good idea, and hopefully would lead to an exchange of useful information and tips; and maybe, the formation of little groups of members, who would benefit from the gatherings and could feed back to us at "HEADQUARTERS" the interesting and creative products of their endeavours; for the further benefit of the other members. So if you would like to have your name and address published, write in to us, marking your envelope for the attention of ROY SMITH.

THE SOFTWARE LIBRARY SERVICE IS FOR MEMBERS ONLY

LIBRARY SOFTWARE TITLES

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

Games

SHOOT

by Nigel Haslock – Switzerland.

Shoot down aircraft and helicopters, using your joystick controlled cannon! This program, written in BASIC, produces a machine code boot tape which runs with no cartridge. Novel feature is it can duplicate itself.

Runs in 16K Cassette minimum.

COLLISION COURSE

by Jon Beff – Manchester.

Avoid collisions with your opponent, but try to trap him into colliding with you. For two players or against the computer.

Runs in 16K Cassette or Disk minimum.

YAHTZEE

by Steve Calkin – Basildon.

Dice game, where you have to get two, three or four of a kind, etc.

Runs in 16K Cassette or 32K Disk minimum.

SKYBLITZ VERSION 1

by Chris Barlow – Leigh.

Drop bombs from your spacecraft onto the buildings below, reduce them to rubble before you fly too low and crash into them.

Runs in 16K Cassette or Disk minimum.

SUBMARINE HUNTER

by Hugh Denholm – Aberdeen.

Drop bombs from your helicopter to try and sink the sub, avoiding missiles from the protecting destroyer.

Runs in 16K Cassette or 32K Disk minimum.

COWBOY

by Evan Fraser – Edinburgh.

Shoot your partner three times to kill him, reload with bullets from the local store.

Runs in 32K Cassette or Disk minimum.

BATTLE OF BRITAIN

by Mike Barnard – Guisborough.

A strategic game of wits, defending Britain against wave after wave of attacking bombers.

Runs in 16K Cassette or Disk minimum.

FRUIT MACHINE

by Mike Nash – Radstock.

A graphic representation of a fruit machine, where you can gamble the "computers money". Incorporates nudge and hold facilities.

Runs in 32K Cassette or Disk minimum.

SKYBLITZ VERSION 2

by Mike Barnard – Guisborough.

New version of Skyblitz 1, with improved graphics, sound and joystick control.

Runs in 16K Cassette or Disk minimum.

MOONLANDER

by D. Mensing – Sutton Coldfield.

Manoeuvre your craft onto the landing pads.

Runs in 16K Cassette or 32K Disk minimum.

COUNTDOWN

by P. Stevens – Horley.

Hit moving targets with a bouncing ball and joystick controlled bats.

Runs in 16K Cassette or 32K Disk minimum.

HANGMAN

by R. L. Howarth – Preston.

Save the man from the gallows by guessing the word.

Runs in 16K Cassette or 32K Disk minimum.

MANIAC DRIVER

by P. J. Phillips – Sevenoaks.

Avoid oncoming traffic by skilful driving. This game uses PADDLES.

Runs in 16K Cassette or 32K Disk minimum.

LIBRARY SOFTWARE

PHANTOM FLAN FLINGER
by Chris Barlow – Leigh.

Throw flans into the dodging face and score points.

Runs in 16K Cassette or 32K Disk minimum.

COLOUR SNAP
by H. Clark – Barking.

ATARI version of the popular card game.

Runs in 16K Cassette or Disk minimum.

INVADERS
by Anthony Ball – Preston.

Although this version of space invaders is written in BASIC, it still retains the addictive quality of the original.

Runs in 48K Cassette or Disk (note this program will run in 16K Cassette or 32K Disk, but the titles may be corrupted).

FOUR IN A ROW
by R. P. Bosma – Canterbury.

Drop your marker in the grid and stop your opponent from getting four in a row.

Runs in 16K Cassette or Disk minimum.

DOGFIGHT
by Rod Knowles – Merseyside.

A game for two players involving World War I bi-planes in combat.

Runs in 32K Cassette or Disk minimum.

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.

PEDESTRIAN
by P. Stevens – Horley.

You are the pedestrian and you must cross the road without getting run over.

Runs in 16K Cassette or 32K Disk minimum.

ASTRO DODGER
by D. Dodson – Leigh-on-Sea.

Dodge between the asteroids to destroy the enemy craft.

Runs in 16K Cassette or 32K Disk minimum.

REVERSI
by Ian Finlayson – Gosport.

BASIC version of this two player game with running scores.

Runs in 16K Cassette or Disk minimum.

CONNECT 4
by R. W. Askew – Northampton.

Use cunning and skill to stop your opponent from connecting four.

Runs in 16K Cassette or 32K Disk minimum.

GALACTIC CUBE
by Nigel Haslock – Switzerland.

Stear your craft to safety out of the space cube.

Runs in 16K Cassette or Disk minimum.

SKYBLITZ VERSION 4
by Jon Beff – Manchester.

Superb adaptation from the original, includes redefined character set to give enhanced graphic presentation.

Runs in 16K Cassette or 32K Disk minimum.

MOLE
by Keith Mayhew – Rayleigh.

Stop the moles from digging up your garden by hitting them on the head.

Runs in 16K Cassette or Disk minimum.

Adventure Games

STONEVILLE MANOR
by Nigel Haslock – Switzerland.

Extensive BASIC word adventure, the object of which is to discover the treasures hidden in Stoneville Manor. Unfinished games can be saved onto cassette (slight modification of the program allows you to save to disk).

Runs in 32K Cassette or Disk minimum.

THE VALLEY
by Steve Calkin – Basildon.

Semi-graphic adventure, you can be warrior, wizard, priest, etc. and you must fight your way to safety along the forest path.

Runs in 32K Cassette or Disk minimum.

OUTPOST
by Anthony Ball – Preston.

Graphic adventure in which you defend the outpost from attacking enemies of varying strengths.

Runs in 32K Cassette or 48K Disk minimum.

Home Entertainment

POLYGONS
by Chris Rutter – New Zealand.

Make polygons in Graphics 7 to 11, use your joystick to change the colours.

Runs in 16K Cassette or 32K Disk minimum.

SYNTHESISER
by Chris Payne – Manchester.

Program your keyboard to act as a synthesiser.

Runs in 16K Cassette or 32K Disk minimum.

DUNGEONS & DRAGONS CHARACTER GENERATOR
by A. J. Palmer – Basingstoke.

An absolute must for all dungeons and dragon players.

Runs in 16K Cassette or 32K Disk minimum.

BIORHYTHM
by Ezio Bottarelli – Italy.

Forecast your physical, intellectual and emotional future.

Runs in 16K Cassette or Disk minimum.

LIE DETECTOR
by D. Dodson – Leigh-on-Sea.

Instructions are included on how to make the hand held sensors. Then a display is given in the form of a graph showing true & false areas. As you hold the sensors and are asked awkward questions by a friend (?) he/she can see how often you LIE!!!

Runs in 16K Cassette or 32K Disk minimum.

DARTS SCOREBOARD
by Derek Harrison – Glasgow.

Let the computer keep score in your game of darts and give a fanfare to the winner.

Runs in 32K Cassette or Disk minimum.

NOUGHTS & CROSSES
by Ken Hall – Okehampton.

ATARI version of the popular game.

Runs in 16K Cassette or 32K Disk minimum.

Demo's

SPIRAL
by Nigel Haslock – Switzerland.

Draw spirals of differing patterns depending on user input.

Runs in 16K Cassette or 32K Disk minimum.

ATARI CLOCK
by Ian Lawson-Smith – Watford.

An alarm clock for your home computer.

Runs in 16K Cassette or 32K Disk minimum.

PLAYER MISSILE DEMO
by Keith Mayhew – Rayleigh.

A step by step demonstration of how to create player missiles on the ATARI.

Runs in 16K Cassette or Disk minimum.

256 COLOURS
by Keith Mayhew – Rayleigh.

A short program which will display all 256 colours available on the ATARI on the screen at once.

Runs in 16K Cassette or 32K Disk minimum.

COLOUR CORRIDOR
by Keith Mayhew – Rayleigh.

See the colours roll down the corridor.

Runs in 16K Cassette or 32K Disk minimum.

MEMORY SCROLLER
by Keith Mayhew – Rayleigh.

Scroll through memory a page at a time.

Runs in 16K Cassette or Disk minimum.

ATARI TRAIN
by Keith Berry – Birmingham.

Short demo incorporating player missile graphics.

Runs in 16K Cassette or Disk minimum.

SNOOPY
by Chris Davies – Bromley.

Sketches SNOOPY on the screen in graphics 8.

Runs in 16K Cassette or 32K Disk minimum.

SPHERES
by Peter Patay – Oxted.

Draws random spheres in graphics 9.

Runs in 16K Cassette or 32K Disk minimum.

QUADRANTS
by Peter Patay – Oxted.

A random pattern is generated in four positions to give kaleidoscopic effect.

Runs in 16K Cassette or Disk minimum.

PICASSO & PYTHAGORAS
by H. Clark – Barking.

Artistic patterns created by Pythagorian triangles.

Runs in 16K Cassette or Disk minimum.

PROBLEM & SOLUTION
by Ian Finlayson – Gosport.

A problem is set and a solution is given. Can you write a better program to solve the problem?

Runs in 16K Cassette or Disk minimum.

STERLING
by Allan Sharpe – Burgess Hill.

Character redefinition program, replaces "&" with a pound symbol.

Runs in 48K Cassette or Disk minimum.

ART-6
by R. P. Bosma – Canterbury.

Six artistic demo's in Graphics 8.

Runs in 16K Cassette or Disk minimum.

U.S.S. ENTERPRISE
by Alex Kells – Liverpool.

Shows what can be achieved using simple Graphics 8 techniques.

Runs in 16K Cassette or 32K Disk minimum.

Utilities

FILER 1
by Chris Payne – Manchester.

A filing system for cassette owners.

Runs in 16K Cassette minimum.

WEDGE
by Anthony Ball – Preston.

This program is for disk drive owners and it writes an AUTORUN.SYS file and adds commands to BASIC, such as KILL, RENAME, LOCK, UNLOCK, DIR, SCRATCH, without entering DOS.

Runs in 32K Disk minimum.

LIBRARY SOFTWARE

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

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

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

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

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

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

TOP TEN

1	SHOOT	NIGEL HASLOCK
2	CHARACTER GENERATOR 1	MARTIN WALKER
3	THE VALLEY	STEVE CALKIN
4	ASSEMBLER	CHRIS RUTTER
5	SYNTHESISER	CHRIS PAYNE
6	STONEVILLE MANOR	NIGEL HASLOCK
7	FRUIT MACHINE	MIKE NASH
8	256 COLORS	KEITH MAYHEW
9	MOONLANDER	D. MENSING
10	MEMORY SCROLLER	KEITH MAYHEW

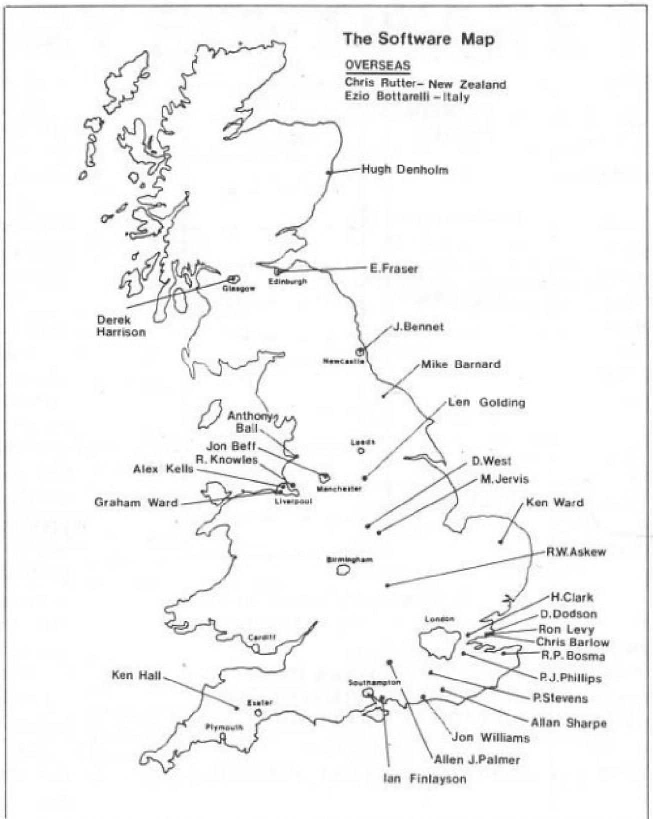
These are the top ten requested exchange programs, since the last issue of the newsletter. "SHOOT" was the most requested, and in fact outstripped the title in second place by nearly 50%. The two adventure games "VALLEY" and "STONEVILLE MANOR" proved very popular; obviously there are a lot of people out there who spend long nights fighting against evil monsters and fearsome creatures (I'm sure there would be a big market for an adventure game called "MOTHER-IN-LAW"). Keith Mayhew's "256 COLORS" has moved into the chart at No. 8, I think people must be trying to find out if the ATARI really can display 256 colours at once. P.S. it can!

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.

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

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

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



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

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

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

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

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

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

Education

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

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

Music

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

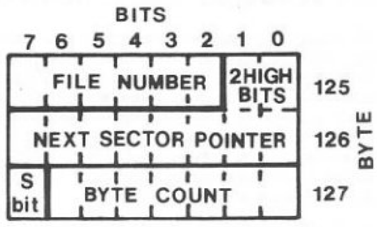
SECTOR – THE MISSING LINK

by Ron Levy

The idea of "SECTOR" is to allow those of you who own an ATARI 810 disk drive to experiment, without being limited by DOS to the file structure. With SECTOR you are able to load, edit, and save *any* sector on the disk. With a reasonable understanding of DOS II's file structure, you can perform all kinds of "nifty" things, such as retrieving deleted files and repairing damaged files. Examining and altering auto-boot disks is also greatly simplified. As an understanding of the way data is stored by DOS II will be of help, I will briefly outline its file structure.

The 810 drive organises the floppy disk as a collection of numbered blocks of bytes called "SECTORS". There are 720 sectors or blocks, and each holds 128 bytes or characters. As each file is created, an empty block is found and the data is poured into it. The problem soon occurs when the sector is filled, for another free sector has to be allocated and somehow linked to the first, so that when the file is being read the second sector can be found. As I will explain later, the directory information for this file only tells the DOS where to find its first sector. So how does DOS find the rest of the file? What actually happens is that as the file was originally being created only 125 of the 128 bytes in each block are used for the USERS data, and the remaining 3 bytes are kept and used by DOS to provide three functions: (a) to "point" to the next sector in the file, (b) to say which number file the sector belongs to, (c) to indicate if the sector is a "short" sector and, if so, how many bytes are valid.

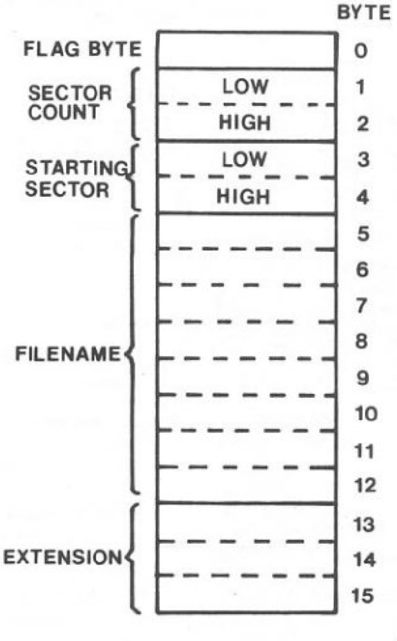
The first (a) is obviously the key to the way in which DOS finds the next sector previously allocated to the file. The second function (b) is not really essential, but it is useful, because as DOS was creating the file it notes the occurrence of the files name entry in the directory and places this number into one of the last three bytes of each sector used by the file. When ever the file is read back, if there is ever a discrepancy between the value of this byte and the directory occurrence, DOS assumes there has been some problem. It will report this to the user as the dreaded "ERROR 164", file number mismatch. This unhappy event is usually caused by the careless user either "BREAK"ing or "SYSTEM RESET"ing during a disk operation, or swapping disks in a drive while a file is still open on the drive. Both are !!**!!* mistakes, which should be avoided at all costs! The last function (c) is a vital one, for a file may not have used all of the bytes in its last sector, and if this is the case DOS needs to know this fact and how many bytes of that sector are allocated to the file. There are 3 functions and 3 bytes, so it would seem logical to have one byte per function. This can not be so, however, because there are 720 sectors on a disk, and so more than one byte is required to store the "NEXT SECTOR" information. Since the directory position byte number does not have to be larger than 63 it does not require all 8 BITS of its byte, so two of its bits are used by the next sector pointer.



This is how the sectors bytes are allocated. Looking at the diagram you will see that one of the BITS has not been explained yet, the "S" bit on byte 127. If the last sector of a file is not completely used, then the "S" bit is set to logic high, and the "BYTE COUNT" will give the actual number of valid bytes.

THE DIRECTORY

There are 8 sectors (361-368) allocated to the disk directory, each holding 8 entries, i.e. 64 entries total. The 16 bytes of each file directory are allocated as follows:



The flag byte is used to indicate the status of the file and the bits are mapped as follows:

BIT No * IF SET HIGH, THEN
BIT 7 * FILE HAS BEEN DELETED
BIT 6 * FILE ENTRY EXISTS
BIT 5 * FILE IS LOCKED
BIT 0 * FILE IS OPEN FOR OUTPUT

Thus the flag byte may have, or be given, the following values:

VALUE * STATUS
$00 * ENTRY NOT YET USED
$40 * ENTRY IS IN USE (normal closed file)
$41 * ENTRY IS IN USE (and file is currently open for output)
$60 * ENTRY IS IN USE (AND file is locked)
$80 * ENTRY IS AVAILABLE (PRIOR file has been deleted)

The SECTOR COUNT (number of sectors in the file) and the STARTING SECTOR NUMBER are obvious, as is the filename. Note, however, that DOS does not insert the full stop before the extension, the directory manager routines remove and insert this for the user's convenience.

THE PROGRAM

The precise format of the program is important, so be careful to include the correct number of spaces and characters where applicable, otherwise you may find some strange numbers will result. The program, when first RUN, will take over 10 seconds to initialise its string and arrays, so if you BREAK out of the program you can resume it by typing GOTO 100. This continues without having to re-initialise! The program is based around a menu, and has 5 options!

1. **Normal Directory Listing.** This gives the standard disk directory listing in two column format. Typing RETURN will take you back to the menu, any other key then RETURN will re-run the directory.

2. **Load Sector.** This allows you to load a sector into the buffer, and will first ask you for the sector number in "HEX", i.e. 001 to 2D0. Typing RETURN only will cause the program to ask for a decimal value in the range 1 to 720. A further RETURN will allow you the option of loading the current sector number by typing '*'.

3. **Save Sector.** This is identical to the load sector option in its use.

4. **Edit Sector.** This is the major section of the program. The current buffer contents are displayed in the form of a matrix, and there are several options available. These are (a) pressing START key aborts the matrix display and asks for the X and Y co-ordinates to edit, (b) pressing OPTION suppresses the printing of the hexadecimal buffer listing, but still gives the character, (c) pressing SELECT suppresses the printing of the character buffer listing, but still gives the hexadecimal byte value table.

When the program asks for the X co-ordinate to edit, the following commands are available:

a) "+" = load and display the next disk sector.

SECTOR

b) "-" = load and display the previous disk sector.

c) "N" = load and display the next sector in the same file as the current sector (if valid).

d) "P" = dump the display to a printer.

If none of these options are required, then you can either type RETURN to get back to the main menu, or type the X co-ordinate of the byte you wish to alter. You will then be asked for the Y co-ordinate, after which you can insert the hex or decimal number, or an ASCII string.

5. **Examine Directory Sectors.** This allows you to examine the disks directory sectors directly. It prints out the flag byte, the number of sectors, and the starting sectors for each file entry to the screen. If the START key is held down whilst this is entered, everything is printed to the printer rather than the T.V. screen. Pressing the START key after the routine has been entered will pause the output to screen or printer. Pressing SELECT will restart at the first directory sector, and pressing OPTION will return you to the main menu.

USING SECTOR

Quite apart from simply experimenting and learning about the Disk System, there are many practical uses for "SECTOR." For instance, if you have accidentally deleted an irreplaceable file on a disk, it can be retrieved by finding its old directory allocation using option 5 (Examine Directory Sectors), then using the sector edit facility to alter the FLAG byte to $40. You should then copy off to a fresh, formatted diskette all the wanted files.

One important part of the "ATARI DISK SYSTEM" is the "VOLUME TABLE of CONTENTS)) sector (sector 360) in which DOS keeps track of which sectors are in use and which are free for new or extended files. I will cover this subject and many others in a more detailed study of the disk system in another issue of this newsletter. Meanwhile I hope you have many interesting hours of experimentation.

Runs in 32K or 48K Disk Systems

```
0 REM .           SECTOR
1 REM .  A Disk Utility Programme.
4 REM .         By Ron Levy.
5 BUFF=1536:UNIT=1:POKE 709,15
10 DIM F$(30),OPT$(10),HEX$(512),T$(16),NVAL$(30),X$(4),
Y$(4),SECTOR$(10),DISK$(5),B(7),BYT$(256)
12 FOR X=1 TO 5:READ Y:DISK$(X,X)=CHR$(Y):NEXT X
18 ? "STAGE 1"
19 REM <<< SET UP HEX CONVERTER >>>
20 T$="0123456789ABCDEF"
22 FOR X=1 TO 256
24 Y=INT((X-1)/16):Y2=X-Y*16:Y=Y+1
26 L=LEN(HEX$)+1:HEX$(L,L)=T$(Y,Y)
27 L=LEN(HEX$)+1:HEX$(L,L)=T$(Y2,Y2)
28 NEXT X
30 C=53279:REM Consol Switches.
39 ? "STAGE 2"
40 REM << Set Up Character Array >>
41 FOR BYT=1 TO 255:BYT$(BYT,BYT)=CHR$(BYT)
44 IF BYT>26 AND BYT<32 OR BYT>124 AND BYT<128 OR BYT>15
4 AND BYT<160 OR BYT>252 THEN BYT$(BYT,BYT)=CHR$(0)
46 NEXT BYT
50 REM <<<< Create Display List >>>>
52 D=561:D0=PEEK(D):D1=D0-1:DL=PEEK(560)+D0*256:DL1=DL-2
56
53 FOR A=1 TO 6:POKE DL1+A,PEEK(DL+A):NEXT A
54 FOR A=6 TO 50 STEP 2:POKE DL1+A,0:POKE DL1+1+A,2:NEXT
 A
55 FOR A=52 TO 53:POKE DL1+A,PEEK(DL+A-23):NEXT A
56 POKE DL1+54,PEEK(561)-1
100 REM <<<<<<<<< Main Menu. >>>>>>>>
102 GRAPHICS 0:POKE 709,15:CMD=0
105 ? "          Sector Utility."
106 ? "          ================";?
108 ? "             By Ron Levy.";?
112 ? "     Disk Directory ...... <1>"
114 ? "     Load Sector ......... <2>"
116 ? "     Save Sector ......... <3>"
118 ? "     Edit Sector ......... <4>"
120 ? "     Examine Directory ... <5>"
150 POSITION 13,20:? "Option -->";
160 CLOSE #2:OPEN #2,4,0,"K:":GET #2,X:CLOSE #2
170 X=X-48:IF X<1 OR X>5 THEN 100
180 ON X GOTO 1000,2000,3000,4000,5000
```

```
200 FOR I=1 TO SIZE
1000 REM <<<<<< DISK DIRECTORY >>>>>>
1020 GRAPHICS 0:POKE 709,15
1100 TRAP 1600:OPEN #1,6,0,"D:*,*"
1110 INPUT #1,F$:? F$;"  ";
1120 INPUT #1,F$:? F$:GOTO 1110
1600 CLOSE #1:INPUT X$:IF X$="" THEN 100
1630 GOTO 1000
2000 REM <<<<<<< Load Sector >>>>>>>
2100 GRAPHICS 0:POKE 709,15:PRINT :TRAP 40000
2110 ? "      LOAD Sector Routine.":?
2200 ? " Which Sector <HEX> --> ";
2210 INPUT SECTOR$:IF SECTOR$="" THEN PRINT :GOTO 2400
2220 IF LEN(SECTOR$)<>3 THEN 2000
2230 I=ASC(SECTOR$(1,1)):J=ASC(SECTOR$(2,2)):K=ASC(SECTO
R$(3,3))
2240 IF I<65 THEN I=I-48:GOTO 2260
2250 I=I-55
2260 IF J<65 THEN J=J-48:GOTO 2280
2270 J=J-55
2280 IF K<65 THEN K=K-48:GOTO 2300
2290 K=K-55
2300 SECT=I*256+J*16+K
2310 IF SECT<1 OR SECT>5720 THEN ? CHR$(253):GOTO 2000
2330 SECTOR=SECT:GOSUB 10000:GOTO 100
2400 ? " Which Sector <DEC> --> ";
2410 INPUT SECTOR$:IF SECTOR$="" THEN PRINT :GOTO 2500
2430 TRAP 2400:SECT=VAL(SECTOR$):TRAP 40000
2440 IF SECT<1 OR SECT>720 THEN ? CHR$(253);:GOTO 2400
2450 SECTOR=SECT:GOSUB 10000:GOTO 100
2500 ? "TYPE * TO LOAD SECTOR ";SECTOR;" ";
2520 INPUT SECTOR$:IF SECTOR$<>"*" THEN 100
2530 ? :? "Ok. -- LOADing Now....";
2550 GOSUB 10000:GOTO 100
3000 REM <<<<<<< Save Sector >>>>>>>
3100 TRAP 40000:CMD=1:GRAPHICS 0:POKE 709,15:PRINT
3110 ? "      SAVE Sector Routine.":?
3200 ? " Which Sector <HEX> --> ";
3210 INPUT SECTOR$:IF SECTOR$="" THEN PRINT :GOTO 3400
3220 IF LEN(SECTOR$)<>3 THEN 3000
3230 I=ASC(SECTOR$(1,1)):J=ASC(SECTOR$(2,2)):K=ASC(SECTO
R$(3,3))
3240 IF I<65 THEN I=I-48:GOTO 3260
3250 I=I-55
```

```
3260 IF J<65 THEN J=J-48:GOTO 3280
3270 J=J-55
3280 IF K<65 THEN K=K-48:GOTO 3300
3290 K=K-55
3300 SECT=I*256+J*16+K
3310 IF SECT<1 OR SECT>5720 THEN ? CHR$(253):GOTO 3000
3320 SECTOR=SECT:GOSUB 10000:GOTO 100
3340 GOTO 100
3400 ? " Which Sector <DEC> --> ";
3410 INPUT SECTOR$:IF SECTOR$="" THEN 3500
3430 TRAP 3400:SECT=VAL(SECTOR$):TRAP 40000
3440 IF SECT<1 OR SECT>720 THEN ? CHR$(253);:GOTO 3400
3450 SECTOR=SECT:GOSUB 10000:GOTO 100
3500 PRINT
3510 ? :? "Type * To SAVE Sector ";SECTOR;" ";
3520 INPUT SECTOR$:IF SECTOR$<>"*" THEN 100
3530 ? :? "Ok. -- SAVEing Now....";:GOSUB 10000:GOTO 100
4000 REM <<<<<<< EDIT SECTOR >>>>>>>
4040 GRAPHICS 0:POKE 709,15:POKE D,D1:TRAP 40000
4050 ? "Sector>";:PRINT SECTOR
4055 POSITION 14,0:? "File>";
4060 BYTE=PEEK(BUFF+125):GOSUB 11000
4065 FILE=(BYTE-B(1)*2-B(0))/4:? FILE;
4070 NXTSEC=PEEK(BUFF+126)+256*(B(1)*2+B(0))
4074 IF PEEK(BUFF+127)>127 THEN NXTSEC=0
4075 POSITION 23,0:? "Next Sec.>";NXTSEC
4080 POSITION 8,1:? "$";
4085 X=INT(SECTOR/256):? HEX$(X*2+2,X*2+2);:X=SECTOR-X*2
56:? HEX$(X*2+1,X*2+2);
4090 POSITION 18,1:? "$";HEX$(FILE*2+1,FILE*2+2);
4100 POSITION 32,1:? "$";:X=B(1)*2+B(0)
4105 ? HEX$(X*2+2,X*2+2);
4110 Y=PEEK(BUFF+126):? HEX$(Y*2+1,Y*2+2)
4160 ? "  0  1  2  3  4  5  6  7  01234567";
4170 POSITION 3,3:? CHR$(17);
4180 FOR X=1 TO 8
4184 ? CHR$(18);CHR$(18);CHR$(32);
4186 NEXT X:? "--------";
4200 FOR L=0 TO 15 STEP 1
4210 POSITION 1,L+4:? L;
4220 POSITION 3,L+4:? CHR$(124);
4248 IF PEEK(C)=5 THEN 4309
4249 POSITION 4,L+4
4250 FOR PK=0 TO 7
4270 BYT=PEEK(BUFF+PK+L*8)
4280 ? HEX$(BYT*2+1,BYT*2+2);" ";
4290 NEXT PK
4300 REM ... PRINT CHARACTERS ...
4305 IF PEEK(C)=3 THEN 4400
4309 POSITION 28,L+4
4310 FOR CH=0 TO 7
4320 BYT=PEEK(BUFF+CH+L*8)
4330 IF BYT=0 THEN ? CHR$(0);:GOTO 4390
4350 PRINT BYT$(BYT,BYT);
4390 NEXT CH
4395 IF PEEK(C)=6 THEN 4500
4400 NEXT L
4500 REM ...EDIT-IT...
4510 ? :POSITION 4,21:? "X Co-ord.->";
4516 INPUT X$:IF X$="" THEN 100
4517 IF X$="N" THEN SECTOR=NXTSEC:CMD=0:GOSUB 10000:GOTO
 4000
4518 IF X$="P" THEN GOSUB 12000:CLOSE #1:GOTO 4000
4519 IF X$="+" THEN SECTOR=SECTOR+1:CMD=0:GOSUB 10000:GO
TO 4000
4520 IF X$="-" THEN SECTOR=SECTOR-1:CMD=0:GOSUB 10000:GO
TO 4000
4522 POSITION 4,22:? "Y Co-ord.-->";:INPUT Y$:IF Y$="" T
HEN 100
4524 TRAP 4500:X=VAL(X$):Y=VAL(Y$):TRAP 40000
4526 IF X<0 OR X>7 OR Y<0 OR Y>15 THEN 4510
4560 EDBYT=Y*8+X:POSITION 23,21:? "Seq= ";EDBYT
4570 POSITION 4,22:? "                       ";
4580 POSITION 4,22:? "New Value <HEX>->";
4590 INPUT NVAL$:IF NVAL$="" THEN 4700
4600 J=ASC(NVAL$(1,1)):K=ASC(NVAL$(2,2))
4610 IF J<65 THEN J=J-48:GOTO 4630
4620 J=J-55
4630 IF K<65 THEN K=K-48:GOTO 4660
4650 K=K-55
4660 NBYT=J*16+K
4670 ? NBYT,HEX$(NBYT*2+1,NBYT*2+2)
4680 POKE BUFF+EDBYT,NBYT:GOTO 4000
4700 POSITION 4,22:? "New Value <DEC>->";
4710 INPUT NVAL$:IF NVAL$="" THEN 4800
4720 TRAP 4700:NBYT=VAL(NVAL$)
4730 POKE BUFF+EDBYT,NBYT:TRAP 40000:GOTO 4000
4800 POSITION 4,22:? "                       ";
4805 POSITION 4,22:? "New <STRING> ";
4810 INPUT NVAL$:IF NVAL$="" THEN 100
4820 FOR X=1 TO LEN(NVAL$)
4830 POKE BUFF+EDBYT+X-1,ASC(NVAL$(X,X))
4850 NEXT X:GOTO 4000
5000 REM <<<< Examine Directory >>>>
5020 GRAPHICS 0:POKE 709,15:TRAP 40000:CNT=0:CLOSE #1
5030 IF PEEK(C)=7 THEN OPEN #1,8,0,"E:":POKE 709,15:P=0:
POKE D,D1:GOTO 5100
5040 TRAP 100:OPEN #1,8,0,"P:":TRAP 40000:P=1
5050 IF PEEK(C)<>7 THEN 5050
5060 P=1
5100 CNT=0:FOR I=0 TO 7:REM <<<Sectors.>>>
5110 SECTOR=I+361:CMD=0:GOSUB 10000
5140 ? #1:? #1;"Sector ";SECTOR;" <$";
5145 X=PEEK(779)*2+2:Y=(SECTOR-INT(SECTOR/256)*256)*2
5150 ? #1;HEX$(X,X);HEX$(Y+1,Y+2);">   ";;
5155 ? #1;"Seq No: ";I;" <$";HEX$(I*2+1,I*2+2);">"
5170 ? #1;"No: Flag Sects Start"
5175 ? #1;"--  ---- ----- -----"
5200 FOR J=0 TO 127 STEP 16
5205 ? #1;HEX$(CNT*2+1,CNT*2+2);"    ";
5210 X=PEEK(BUFF+J):FLAG=X
5220 ? #1;HEX$(X*2+1,X*2+2);"   ";
5230 X=PEEK(BUFF+J+2)
5240 ? #1;HEX$(X*2+2,X*2+2);
5250 X=PEEK(BUFF+J+1)
5260 ? #1;HEX$(X*2+1,X*2+2);"   ";
5270 X=PEEK(BUFF+J+4)
5280 ? #1;HEX$(X*2+2,X*2+2);
5290 X=PEEK(BUFF+J+3)
5300 ? #1;HEX$(X*2+1,X*2+2);"  ";
```

SECTOR

```
5390 X=0:IF FLAG>127 THEN X=1:FLAG=0
5400 FOR S=BUFF+J+5 TO BUFF+J+12
5405 Y=PEEK(S):IF X THEN IF P=0 THEN Y=Y+128
5410 ? #1;CHR$(Y);
5450 NEXT S
5460 ? #1;".";
5470 FOR S=BUFF+J+13 TO BUFF+J+15
5480 Y=PEEK(S):IF X THEN IF P=0 THEN Y=Y+128
5485 PRINT #1;CHR$(Y);
5490 NEXT S:IF X THEN ? #1;" ***";
5500 IF FLAG>95 THEN ? #1;" *";
5600 CNT=CNT+1:? #1:IF PEEK(C)<>7 THEN GOSUB 5950
5650 NEXT J:NEXT I
5800 ? #1;"    Re-run <Y> -->";
5810 INPUT X$:IF X$="Y" THEN 5000
5900 CLOSE #1:GOTO 100
5949 REM << A Little Delay! >>
5950 IF PEEK(C)=6 THEN 5950
5955 IF PEEK(C)=3 THEN POP :CLOSE #1:GOTO 100
5960 IF PEEK(C)=6 THEN RETURN
5961 IF PEEK(C)=5 THEN POP :GOTO 5100
5965 IF PEEK(C)=7 THEN 5955
10000 REM <<<<< Disk Interface >>>>>
10100 POKE 769,UNIT
10110 IF CMD=0 THEN POKE 770,82
10120 IF CMD=1 THEN POKE 770,87
10130 POKE 772,BUFF-INT(BUFF/256)*256
10140 POKE 773,INT(BUFF/256)
10150 POKE 778,SECTOR-INT(SECTOR/256)*256
```

```
10160 POKE 779,INT(SECTOR/256)
10300 A=USR(ADR(DISK$))
10400 IF PEEK(771)=1 THEN RETURN
10410 ? :? CHR$(253):? "ERROR --> ";PEEK(771)
10430 ? "Type <*> To Re-try ... ";
10450 INPUT X$:IF X$="*" THEN 10300
10470 RETURN
10999 REM
11000 REM <<<<<< Bit Map Calc. >>>>>>
11100 FOR IT=0 TO 7:B(IT)=0:NEXT IT:B=BYTE
11110 IF B>127 THEN B=B-128:B(7)=1
11120 IF B>63 THEN B=B-64:B(6)=1
11130 IF B>31 THEN B=B-32:B(5)=1
11140 IF B>15 THEN B=B-16:B(4)=1
11150 IF B>7 THEN B=B-8:B(3)=1
11160 IF B>3 THEN B=B-4:B(2)=1
11170 IF B>1 THEN B=B-2:B(1)=1
11180 IF B>0 THEN B(0)=1
11900 RETURN
12000 REM <<<<< Printer Routine >>>>>
12010 TRAP 12900:OPEN #1,8,0,"P:":TRAP 40000
12020 FOR Y=0 TO 20:POSITION 0,Y
12030 FOR X=0 TO 39:GET #6,A:IF A<32 OR A>122 THEN A=32
12040 PUT #1,A:NEXT X:? #1:NEXT Y:? #1:CLOSE #1:RETURN
12900 ? CHR$(253),"ERROR ->";PEEK(851)
12910 ? "<RETURN> To Continue ..";:INPUT OPT$:RETURN
30000 DATA 104,32,83,228,96
32767 END
```

PEDESTRIAN

Runs in 16K Cassette or 32K Disk

by Paul Stevens — Horley

This game, although written in BASIC, if seen on a lesser machine than an ATARI, you would assume it was written in machine code. The concept of the game is similar to "CHICKEN" or "PREPPIE" or "FROGGER" but you are a pedestrian who has to cross the motorway, avoiding getting squashed by the passing traffic, in order to gain points and progress to the next level. There are 19 levels with bonus points awarded at various stages of the game. As the levels increase so does the speed of the traffic thus making it more difficult to cross. The program makes use of the player missile graphics capabilities of the ATARI to produce some very pleasing visual effects. Sound routines are also incorporated to complement the visual display.

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

```
0 REM PEDEST.BAS - PEDESTRIAN
1 REM Paul Stevens
2 REM 1 Greenfields Close,
3 REM Horley, Surrey, RH6 8HX.
10 CLR :DIM A$(140),B$(12):GRAPHICS 0:POKE 752,1:POKE 70
8,58:POKE 710,224:POKE 711,138:A=PEEK(560)+PEEK(561)*256
20 POKE A+3,71:POKE A+8,6:POSITION 5,0:? "PEDESTRIAN":P
OKE 87,1:POSITION 4,5:? #6;"by p stevens"
30 POKE 87,0:POSITION 13,8:? "INSTRUCTIONS ?":POSITION 1
5,10:? "( Y or N )"
40 IF PEEK(764)=43 THEN 70
50 IF PEEK(764)=35 THEN 200
60 GOTO 40
70 GRAPHICS 0:POKE 708,90:POKE 710,224:POKE 711,58:POKE
A+3,71:POKE A+6,7:POKE 752,1
80 ? "  INSTRUCTIONS":? "  THE OBJECT OF THIS GAME IS TO
 MOVE":? "THE LITTLE MAN FROM THE TOP OF THE"
90 ? "SCREEN TO OFF THE BOTTOM OF THE":? "SCREEN WITHOUT
 BEING HIT BY ANY OF":? "THE CARS."
```

```
100 POSITION 12,7:? " USE JOYSTICK ":? CHR$(29);CHR$(29)
;"  TO PAUSE THE GAME PRESS THE SPACE":? "BAR."
110 ? CHR$(29);"  TO RESUME THE GAME PRESS THE FIRE":? "
BUTTON."
115 ? CHR$(29);"  PRESS SYSTEM RESET TO STOP GAME."
120 ? CHR$(29);"   LEVEL 19 IS THE MAXIMUM SPEED.":POSIT
ION 12,21:? "PRESS ANY KEY":POKE 764,255
130 IF PEEK(764)=255 THEN 130
140 ? CHR$(125):POSITION 6,0:? "scoring":? "  CROSSING T
HE ROAD = 100 POINTS PLUS":? "THE TIME LEFT."
150 ? CHR$(29);CHR$(29);"   CROSSING THE ROAD BEFORE THE
 TIME":? "GETS BELOW 100 = 100 POINTS PLUS THE"
160 ? "TIME LEFT PLUS 200 BONUS POINTS.":? CHR$(29);CHR$
(29);"  RUNNING OUT OF TIME = LOSING A LIFE & NO POINTS.
170 ? CHR$(29);CHR$(29);"  GETTING HIT BY A CAR = LOSING "
 A LIFE& NO POINTS."
175 ? CHR$(29);CHR$(29);"  BONUS LIFE EVERY 1000 POINTS. "
180 POSITION 10,21:? "PRESS START TO PLAY"
```

PEDESTRIAN

```
190 IF PEEK(53279)<>6 THEN 190
200 GRAPHICS 23:POKE 16,64:POKE 53774,64:POKE 559,0:A=PE
EK(560)+PEEK(561)*256:POKE 560,154:POKE 752,1
210 POKE A+3,71:POKE A+6,10:POKE A+7,2:POKE A+98,2:POKE
A+101,6
220 A=PEEK(88)+PEEK(89)*256+3800:FOR B=1 TO 11:READ C,D:
POKE A+C,D:NEXT B
230 POKE 704,14:POKE 705,132:POKE 706,70:POKE 707,54:POK
E 708,8:POKE 709,0:POKE 710,250:POKE 711,138:POKE 712,182
240 POKE 87,2:POSITION 5,0:? #6;"pedestrian":POKE 87,0:P
OSITION 13,1:? "DRAWING ROADS":POKE 559,46:POKE 87,7
250 COLOR 1:FOR A=2 TO 91:PLOT 0,A:DRAWTO 159,A:NEXT A:C
OLOR 3
260 A=1:FOR B=1 TO 3:A=A+21:FOR C=1 TO 2:A=A+1:PLOT 0,A:
DRAWTO 159,A:NEXT C:NEXT B
270 A=12:FOR B=1 TO 4:C=-16:FOR D=1 TO 8:C=C+20:PLOT C,A
:DRAWTO C+10,A:NEXT D:A=A+23:NEXT B:COLOR 0:PLOT 0,95
280 POKE 87,0:POSITION 10,1:? "COMPUTING...10 Secs.":I=P
EEK(106)-24:POKE 54279,I:POKE 53277,3:POKE 623,17
290 FOR A=53257 TO 53259:POKE A,1:NEXT A:POKE 53260,255:
Z=256*I:FOR K=Z+384 TO Z+1024:POKE K,0:NEXT K
300 X=X+1:READ A:IF A<>1 THEN B$(X,X)=CHR$(A):GOTO 300
310 V=PEEK(134)+PEEK(135)*256:A=PEEK(140)+PEEK(141)*256:
G=I*256+512-A:H=INT(G/256):L=G-H*256
320 POKE V+2,L:POKE V+3,H:V=0
330 J=I*256+640:FOR N=23 TO 100:READ A:POKE J+N,A:NEXT N
:RESTORE 380
340 J=I*256+768:FOR N=23 TO 100:READ A:POKE J+N,A:NEXT N
350 J=I*256+896:FOR N=33 TO 110:READ A:POKE J+N,A:NEXT N
360 J=I*256+384:FOR N=22 TO 111:POKE J+N,255:NEXT N
370 DATA 4,226,5,249,7,240,8,206,9,243,10,244,11,229,12,
246,13,229,14,238,15,243,0,24,36,24,255,24,36,36,102,0,0 ,1
380 DATA 224,71,242,255,255,255,242,71,224,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,224,71,242,255,255,255,242,71,224,0,0,0,0
390 DATA 0,0,0,0,0,0,0,0,0,0,224,71,242,255,255,255,242
,71,224,0,0,0,0,0,0,0,0,0,0,0,0,224,71,242
400 DATA 255,255,255,242,71,224
410 DATA 7,226,79,255,255,255,79,226,7,0,0,0,0,0,0,0,0
,0,0,0,0,0,7,226,79,255,255,255,79,226,7,0,0,0,0,0,0,0
420 DATA 0,0,0,0,0,0,7,226,79,255,255,255,79,226,7,0,0,
0,0,0,0,0,0,0,0,0,0,0,7,226,79,255,255,255,79,226,7
430 G=53248:H=53249:I=53250:J=53251:K=53765:L=100:M=190:
N=50:O=3:P=0:Q=1:R=200:T=275:U=0:W=1000:Z=123
440 POSITION 1,1:? "
 "
450 P=P+1:POSITION 1,0:? #6;"                         ":IF P>1
9 THEN P=19
460 POSITION 6,0:? "level ";P
470 POSITION 1,1:? "LIVES = ";O;" ":POSITION 13,1:? "SCO
RE = ";U:POSITION 28,1:? "TIME = ";T
480 FOR A=1 TO 6:FOR B=14 TO 0 STEP -1:POKE 711,B*16+B:S
OUND 0,B*10,10,10:SOUND 1,B*10+6,10,10:NEXT B:NEXT A
490 IF O=0 THEN 910
500 POKE 711,138:POSITION 2,0:? #6;"  pedestrian   ":PO
SITION 9,1:? O;" ":Y=4:POKE 53278,0
510 SOUND 0,R,12,4:SOUND 1,R-10,12,4:POKE 77,0:POKE 764,
255:POKE 53764,25
520 S=STICK(0):IF PEEK(764)=33 THEN 740
530 IF S=14 THEN Y=Y-1:POKE K,196
540 IF S=13 THEN Y=Y+1:POKE K,196
550 IF S=11 THEN Z=Z-1:POKE K,196
```

```
560 IF S=7 THEN Z=Z+1:POKE K,196
570 IF L>218 THEN L=28
580 IF M>218 THEN M=28
590 IF N<28 THEN N=218
600 IF Z<48 THEN Z=48:POKE K,206
610 IF Z>200 THEN Z=200:POKE K,206
620 IF PEEK(53260) THEN 790
630 T=T-1:IF T<0 THEN 780
640 IF Y>124 AND T>=100 THEN 680
650 IF Y>124 AND T<100 THEN 710
660 IF Y<4 THEN Y=4:POKE K,206
670 POSITION 35,1:? T;" ":A$(Y,Y+9)=B$:L=L+Q:M=M+Q:N=N-Q
:POKE G,Z:POKE H,L:POKE I,M:POKE J,N:POKE K,0:GOTO 520
680 POKE K,0:POKE K-4,0:FOR Z=53248 TO 53251:POKE Z,0:NE
XT Z:POSITION 5,0:? #6;"200  bonus":POKE 712,0:POKE 708,0
690 FOR B=1 TO 10:FOR A=0 TO 14:SETCOLOR 2,B,A:SETCOLOR
3,B,A:SOUND 1,A*10,10,8:NEXT A:NEXT B
700 BO=200:Z=123:POKE 708,8:POKE 710,250:POKE 712,182
710 POKE K,0:POKE K-2,0:POKE K-4,0:U=U+100+BO+T:BO=0:IF
P<19 THEN R=R-5:Q=Q+0.25
720 IF U>=W THEN O=O+1:W=W+1000
730 T=275:GOTO 450
740 POKE K,0:POKE K-2,0:POKE K-4,0:POSITION 4,0:? #6;"ga
me  paused":C=0
750 C=C+2:POKE 711,C:IF C=254 THEN C=0
760 FOR D=0 TO 5:IF STRIG(0)=0 THEN POKE 711,138:POSITIO
N 4,0:? #6;" pedestrian ":GOTO 510
770 NEXT D:GOTO 750
780 POKE 711,14:POSITION 2,0:? #6;"your out of time":POK
E 704,0:POKE K,0:GOTO 810
790 POKE 711,15:POSITION 5,0:? #6;"your  dead":POKE 704,
0:POKE K,0
800 A=14:FOR B=250 TO 100 STEP -3:SOUND 0,250-B,8,A:SOUN
D 1,254-B,8,A:NEXT B
810 FOR S=1 TO 11:READ A,B:SOUND 0,A,10,10:SOUND 1,A+2,1
0,10:FOR C=1 TO B:NEXT C:POKE K-2,0:POKE K-4,0:NEXT S
820 DATA 217,75,217,37,217,19,217,75,182,50,193,19,193,5
0,217,19,217,50,225,19,217,75
830 RESTORE 820:SOUND 0,255,10,14:SOUND 1,253,10,14:POKE
 53768,1
840 C=232:FOR S=15 TO 107:C=C-1:POKE 53252,S:POKE 53253,
S+8:IF S+16>Z THEN POKE G,S+16
850 POKE 53255,C:POKE 53254,C-8:IF C-16<Z THEN POKE G,C-
16
860 POKE 711,S:NEXT S:Z=123:POKE K-2,0:GOTO 890:SOUND 0,
255,10,14:SOUND 1,253,10,14:POKE 53768,1
870 C=138:FOR S=107 TO 15 STEP -1:C=C+1:POKE 53253,S+8:P
OKE 53252,S:POKE 53254,C-8:POKE 53255,C:POKE 711,S:NEXT
S
880 POKE 704,14:GOTO 490
890 IF Y<=4 THEN O=O-1:T=275:SOUND 0,255,10,14:SOUND 1,2
53,10,14:POKE 53768,1:GOTO 870
900 Y=Y-1:A$(Y,Y+10)=B$:SOUND 0,Y,8,10:POKE 711,Y:GOTO 8
90
910 POKE K-2,0:POKE K-4,0:FOR Z=53248 TO 53251:POKE Z,0:
NEXT Z:IF U>V THEN V=U
920 C=0:POSITION 1,0:? #6;"high score = ";V:POSITION 1,1
:? "       PRESS START TO TRY AGAIN       "
930 C=C+2:POKE 711,C:IF C=254 THEN C=0
940 FOR E=0 TO 4:IF PEEK(53279)=6 THEN 430
950 NEXT E:GOTO 930
```

Runs in 16K Cassette or 32K Disk

INVADERS

by Anthony Ball — Preston

Probably the most popular computer game to invade our lives since computers became freely accessible to the general public. Here is presented a version of space invaders written in BASIC using redefined characters. We think that a more appropriate name for this game would have been 'Face Invaders'! As all the attacking aliens remind us of smiling faces. Type it in and see if you agree with us!!

The game part of this program runs in 16K; but those of you with less than 48K, may find that part of the introduction titles are corrupted. Don't worry as there is no difficulties with the actual game. Just let the program run through to the next section.

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

```
10 LOOP=0:LOOP2=0:TALLY=0:KILL=0:GOSUB 280:GOTO 140
30 BHG=PEEK(53248):IF BHG>0 THEN GOSUB 210
40 IF LA=1 THEN BG=BG-4:VC$(BG)=BULLET$:POKE 53278,1:IF
BG-4*O<10 THEN VC$(BG,BG+4)=BH$:LA=0
50 BHG=PEEK(53248):IF BHG>0 THEN GOSUB 210
60 IF BF=1 THEN SAW=SAW+3:VC$(SAW)=MISSILE$:IF SAW>102 T
HEN VC$(SAW,SAW+5)=BH$:BF=0:SAW=20
70 IF PEEK(53259)=1 THEN 250
80 IF BF=0 AND TALLY>KILL AND O<16 THEN BF=1:POKE 53255,
DX+3:SAW=14+4*O:TALLY=0
90 DX=DX+4*(STICK(0)=7)-4*(STICK(0)=11):POKE 53248,DX
100 IF STRIG(0)=0 AND LA=0 THEN LA=1:DXM=DX+2:POKE 53252
,DXM:BG=94:VC$(BG)=BULLET$:POKE 77,0
110 TALLY=TALLY+1:RETURN
120 POKE 656,1:POKE 657,2:? "YOU LOSE-PUSH TRIGGER TO PL
AY AGAIN";:SOUND 0,0,0,0:TURNS=0:WAVES=0
130 POKE 656,1:POKE 657,2:POKE 704,PEEK(710)
135 IF STRIG(0)=1 THEN 135
137 POKE 752,1:? CHR$(125)
140 POKE 656,3:POKE 657,2:? "POINTS  ";:SCORE=0:GOSUB 24
0:POKE 705,123:POKE 706,121:KILL=80:HARD=0:HARDER=0
150 POKE 704,119:VC$=BH$:LA=0:BF=0:POKE 53278,1:FOR LOOP
=1 TO 8:FOR LOOP2=80 TO 150 STEP 2:SOUND 0,LOOP2,10,6
155 NEXT LOOP2:NEXT LOOP:TALLY=80:SOUND 0,0,0,0
160 POKE 77,0:KILL=KILL-WAVES*10:IF WAVES>1 THEN HARD=30
:IF WAVES>3 THEN HARDER=60
170 FOR O=1 TO 20:HU=-HU:FOR LOOP=0 TO 8:KZ=PEEK(OE)-HU-
20*(LOOP=8):M=PEEK(XD)-(KZ<0)+(KZ>255)
171 KZ=KZ+256*(KZ<0)-256*(KZ>255)
180 GOSUB 30:POKE XD,M:POKE OE,KZ:GOSUB 30:POKE 756,UJ+4
+2*(PEEK(756)=UJ+4):GOSUB 30+HARD
190 SOUND 0,255,10,15:GOSUB 30+HARDER:SOUND 0,0,0,0:NEXT
 LOOP:NEXT O
200 GOSUB 420:VC$=BH$:LA=0:BF=0:POKE 53278,1:GOTO 120
210 SOUND 0,255,6,10:N=DXM+8*LOOP*(-HU)-64*(HU<0)-48:N=2
*(INT(N/16)):BHG=(BHG=2)+3*(BHG=1)+5*(BHG=8)+7*(BHG=4)-1
211 SOUND 0,255,6,6
220 SCORE=SCORE+10*(10-BHG):SHOTS=SHOTS+1
222 IF SHOTS=24 THEN POP :VC$=BH$:WAVES=WAVES+1:SOUND 0,
0,0,0:GOSUB 420:GOTO 150
225 SOUND 0,255,6,4
230 POSITION N,ABS(BHG):? #6;" "
240 POKE 656,3:POKE 657,8:? " ";SCORE;"     ";:VC$(BG,BG
+2)=BH$:LA=0:POKE 53278,1:SOUND 0,0,0,0:RETURN
250 POKE 706-TURNS,0:POKE 704,PEEK(710):VC$=BH$:FOR O=70
 TO 0 STEP -1:SOUND 0,220,8,O/5:NEXT O:GOSUB 420
260 TURNS=TURNS+1:IF TURNS=3 THEN TURNS=0:SCORE=0:GOTO 120
270 GOTO 150
280 UJ=PEEK(106):IF UJ/2=(INT(UJ/2)) THEN UJ=UJ-8
290 POKE 89,UJ:POKE 88,0:? CHR$(125):POKE 89,UJ+3:? CHR$
(125):POKE 106,UJ:GRAPHICS 18:POKE 756,(UJ+4)
291 FOR DE=1 TO 7:? #6:NEXT DE
293 SETCOLOR 3,3,12
294 SETCOLOR 1,7,12:SETCOLOR 0,11,12
295 ? #6;"     s p a c e":? #6:? #6
296 ? #6;" i n v a d e r s"
300 REM
310 FOR LOOP=8 TO 39:READ N:POKE (UJ+4)*256+LOOP+32,N:NE
XT LOOP
320 FOR LOOP=40 TO 71:READ N:POKE (UJ+6)*256+LOOP,N:NEXT
 LOOP
330 FOR LOOP=72 TO 87:POKE (UJ+6)*256+LOOP-72,0:NEXT LOOP
340 FOR LOOP=88 TO 511:LOOP2=PEEK(57344+LOOP):POKE (UJ+6
)*256+LOOP,LOOP2:POKE (UJ+4)*256+LOOP,LOOP2:NEXT LOOP
350 SOUND 0,0,0,0:GOSUB 500:GRAPHICS 1:POKE 559,34:XD=PE
EK(560)+PEEK(561)*256+5:OE=XD-1
360 POKE 89,UJ+2:POKE 88,128:POKE 752,1:? " ":POKE XD+25
,7:POKE 708,200:POKE 709,198:POKE 710,204:POKE 711,202
370 DIM BH$(128),GF$(1),RR$((INT(ADR(GF$)/1024)+1)*1024-
ADR(GF$)-1),PLAYER$(384),VC$(128),P$(128),PL1$(128),PL2$
(128)
380 DIM MISSILE$(8),BULLET$(8):BH$=CHR$(0):BH$(128)=CHR$
(0):BH$(2)=BH$:P$=BH$:VC$=BH$:BG=88
390 FOR LOOP=1 TO 8:READ O,BHG,HU:P$(97+LOOP,97+LOOP)=CH
R$(O):BULLET$(LOOP,LOOP)=CHR$(BHG):MISSILE$(LOOP,LOOP)=C
HR$(HU):NEXT LOOP
400 PL1$=BH$:PL2$=BH$:PL1$(12)=P$:PL2$=PL1$:POKE 53249,1
85:POKE 53250,171:POKE 704,124:POKE 707,220
410 SAW=10:POKE 53254,100:DX=128:POKE 53248,DX:POKE 5427
9,ADR(PLAYER$)/256:POKE 623,1:POKE 53277,3
420 POKE XD,PEEK(89):POKE OE,128:POKE 559,46:BF=1:SHOTS=
0:HU=-1:POKE 756,UJ+4
430 ? #6;CHR$(125):FOR LOOP2=0 TO 3:POSITION 0,LOOP2*2:F
OR LOOP=1 TO 6:? #6;CHR$(5+32*LOOP2+LOOP2+64*(LOOP2>1));
" ";:NEXT LOOP:NEXT LOOP2
431 RETURN
440 DATA 0,124,214,124,40,108,198,130,0,124,214,124,108,
238,130,238,0,56,130,214,254,186,68,56
445 DATA 0,124,146,182,124,56,0,56,0,124,214,124,40,68,6
8,40,0,124,170,124,108,40,170,238
450 DATA 0,56,124,214,254,198,84,56,0,124,146,218,124,56
,56,0
460 DATA 24,1,0,126,1,0,255,0,0,255,0,128,0,0,64,0,0,128
,0,0,64,0,0,0,0,0,0
500 FOR LOOP=1 TO 7
501 POSITION 0,LOOP-1:? #6;"                    "
510 POSITION 0,LOOP:? #6;"     — ' ' ' ' '"
520 IF BHR=156 THEN BHR=158:GOTO 530
522 BHR=156
530 POKE 756,BHR
535 SOUND 0,255,8,10:FOR TFMB=1 TO 10:NEXT TFMB:SOUND 0,
0,0,0
536 FOR TFMB=1 TO 35:NEXT TFMB
540 NEXT LOOP
550 FOR LOOP=1 TO 500:NEXT LOOP
560 RETURN
```

DISK FILE MANAGER

Runs in 32K or 48K Disk Systems

by D. A. Dodson — Leigh-on-Sea

This program will mainly be of use to those of you who, like myself, have trouble finding a particular disk file when you have several programs on one disk and several disks to search through. As those of you with disks know, there is little room to list all the contents of a disk on the label space. This program, although relatively crude, should alleviate the problem. Basically the program works by reading the directory of a disk and storing this in memory and then transfering the information into a file on a chosen "MASTER DISK".

HOW TO USE

First select a blank formated disk and write DOS files onto it, then type in the program from the listing below and save it off to the disk under a suitable name, i.e. "D:DISKMAN". When you run the program you will be presented with three options, (1) ACCESS FILES, (2) CREATE FILES, (3) DELETE FILES. Option 2 should be pressed first as you obviously need to create files before you can access them.

On pressing 2 you will be given a further two options, MANUAL or AUTOMATIC create. Automatic is for those disks which have a directory, manual for non-directory disks, i.e. autoboot games etc. If you select option 1 (manual) you will be asked to input the filenames of programs on a disk, you can continue entering names followed by return until all programs on that disk have been entered. By pressing return twice you will be asked to allocate a number to this file (any number up to 999). Enter the number and hit return, all the information you have just typed in will be stored in a file on your master disk under the name "DISKFILE.(number)", e.g. "DISKFILE.27", and you will be returned to the main menu to start again.
If you select option 2 (automatic) you will be instructed to remove your master disk and insert into your drive a disk with a directory, and to then press START. The program will then read all the titles on that disk and store them in string arrays.

Once files are loaded you must remove the disk and replace the master disk into the drive so that a file number can be allocated, on hitting return the directory is placed in a file on the master disk. As you record each of your disks you should mark them with the number you have allocated in the diskfile.

Back to the main menu . . . on pressing 1, to ACCESS FILES, you will be given two options, (1) FILE SEARCH (2) LIST ALL. The file search facility is used to find a particular program, you can type in the exact name or if you enter just one letter, all the programs that start with the letter will be indicated. List all obviously does just that, although you do have the option of starting the list from any number you choose.
You can stop the display by holding down SELECT and continue by pressing START.

In the program presented here there is a print option available, but I have also given to the library a version of this program that does not have this option. Finally, once more back at the main menu . . . pressing 3 means you are able to delete files by inputing the number of the file to be erased.

FOOTNOTE:

All DISKFILES created on the master disk are automatically locked. TRAP statements are frequent in order to prevent accidental crashing of the program. Lastly, a plea from the author: if anybody out there knows a way round using all those string array subroutines and/or a way of DIMensioning let me know. My address is 29 BOLNEY DRIVE, LEIGH-ON-SEA, ESSEX.

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

```
 0 REM .              DISK FILE MANAGER
 5 GRAPHICS 0
 10 CLR :CLOSE #1:A=17
 20 DIM B$(11),A$(A),FILE$(14),NAME$(27),TEMP$(A),F$(A)
 25 DIM A1$(A),A2$(A),A3$(A),A4$(A),A5$(A),A6$(A),A7$(A),
A8$(A),A9$(A),A10$(A),A11$(A),A12$(A),A13$(A),A14$(A)
 30 DIM A15$(A),A16$(A),A17$(A),A18$(A),A19$(A),A20$(A),A
21$(A),A22$(A),A23$(A),A24$(A),A25$(A),A26$(A),A27$(A)
 35 DIM A28$(A),A29$(A),A30$(A),A31$(A),A32$(A),A33$(A),A
34$(A),A35$(A),A36$(A),A37$(A),A38$(A),A39$(A),A40$(A)
 40 DIM A41$(A),A42$(A),A43$(A),A44$(A),A45$(A),A46$(A),A
47$(A),A48$(A),A49$(A),A50$(A)
 50 FILE$="D:DISKFILE.":F$(1,1)=CHR$(125):FOR ZZ=2 TO 10:
F$(ZZ,ZZ)=CHR$(29):NEXT ZZ:GOTO 100
 90 POKE 712,162:FOR T=1 TO 5:SOUND 0,0,0,0:FOR T1=1 TO 5
:SOUND 0,20,10,10
 91 NEXT T1:NEXT T:SOUND 0,0,0,0:POKE 712,36:RETURN
 100 REM xxxxMENUxxxxx
 110 POKE 752,1:POKE 710,0:POKE 712,36:OPEN #1,4,0,"K:"
 120 ? CHR$(125):? :? "   PLEASE SELECT OPTION REQUIRED."
:? :?
 130 ? "1 Access diskfiles.":?
 140 ? "2 Create diskfiles.":?
 150 ? "3 Delete disk files.":?
 200 GET #1,K:K=K-48:IF K<0 OR K>3 THEN ? CHR$(253);"NON-
EXISTANT SELECTION.RE-INPUT!!":GOTO 200
 210 CLOSE #1:GOSUB 90:? CHR$(125):ON K GOTO 1000,2500,300
 310 ? F$:? "Please type diskfile No.to be deleted"
 320 TRAP 10:INPUT D:GOSUB 90:FILE$(12)=STR$(D):XIO 36,#1
,0,0,FILE$
```

```
 330 ? F$:? "          DELETING:";FILE$:XIO 33,#1,0,0,FILE$
 340 ? F$:? "     xxxxxxxxxxDONExxxxxxxxxxx":GOSUB 90:FOR
 T=1 TO 500:NEXT T:GOTO 10
 350 ? F$:? CHR$(253);"xxxxNO DISKFILE WITH THAT NUMBERxx
xx":FOR T=1 TO 500:NEXT T:GOTO 310
 500 CLOSE #1:GOSUB 90:? F$:? "   ERROR...Check printer/l
eads....":FOR T=1 TO 1500:NEXT T:GOTO 1000
 1010 OPEN #1,4,0,"K:":? F$:POKE 712,36
 1020 ? "            1 File search.":?
 1030 ? "            2 List all.":?
 1033 ? :? :? "xxxxxHIT SELECT TO STOP LISTINGSxxxxxx"
 1040 GET #1,K:K=K-48:IF K>2 THEN 1040
 1041 ? F$:? "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
 1042 GOSUB 90:? "Please type file number to list from."
 1043 TRAP 1041:? "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
 :INPUT D:D=D-1:? CHR$(125)
 1045 CPY=0:GOSUB 90:? F$:? "Do you require a hard copy,(
Y/N)?";:INPUT B$:IF B$="Y" THEN CPY=1
 1058 ? CHR$(125):GOSUB 90:CLOSE #1:ON K GOTO 1060,1099
 1060 ? F$:POKE 712,36:? "Type first or complete letters
of file":? "              xxxxxxxxxxxxxxx"
 1061 ? "            NO WILD CARDS!.":A=9:INPUT A$:GOSUB 90
 1065 ? CHR$(125):FOR I=1 TO LEN(A$):IF A$(I,I)="." THEN
 A$(I,I)=" ":IF A$(LEN(A$),LEN(A$))<>" " THEN 1075
 1070 B$(I,I)=A$(I,I):NEXT I:GOTO 1099
 1075 FOR T=I TO 8:B$(T,T)=" ":NEXT T
 1080 ? CHR$(125):GOSUB 90:CLOSE #1:ON K GOTO 1060,1099
 1099 CLOSE #1:D=D+1:FILE$(12)=STR$(D)
 1100 TRAP 1200:OPEN #1,4,0,FILE$
```

DISK FILE MANAGER

```
1110 ? :? "    Disk No:";D:?
1115 TRAP 500:IF CPY=1 THEN LPRINT :? "    Disk No:";D:?
1120 TRAP 1099:INPUT #1,NAME$
1130 IF A$>"" THEN 1230
1140 ? NAME$(9,(LEN(NAME$)))
1145 IF CPY=1 THEN LPRINT NAME$(9,(LEN(NAME$)))
1150 IF PEEK(53279)<>5 THEN 1120
1155 GOSUB 90:POKE 712,36:? :? "Hit OPTION for menu.STAR
T to continue."
1160 IF PEEK(53279)=6 THEN GOSUB 90:GOTO 1120
1170 IF PEEK(53279)=3 THEN CLOSE #1:GOSUB 90:GOTO 10
1180 GOTO 1160
1200 ? :? "No diskfile number:";D:IF PEEK(53279)<>5 THEN
 GOTO 1099
1210 GOTO 1155
1230 TEMP$=NAME$(13,LEN(NAME$)-4)
1240 FOR I=1 TO LEN(B$)
1250 IF TEMP$(I,I)=B$(I,I) THEN NEXT I:GOTO 1140
1260 GOTO 1150
2500 REM xxxFILES LOADERxxxx
2505 OPEN #1,4,0,"K:":? F$:? "            1 Manual create."
:? :? "          2 Automatic create."
2510 GET #1,K:K=K-48:IF K<0 OR K>2 THEN ? "INCORRECT INP
UT RE-SELECT..........":GOTO 2510
2515 CLOSE #1:GOSUB 90:ON K GOTO 2700,2520
2520 ? F$:? "Please place disk to be read into":? "Drive
 #1 and hit START............"
2521 POKE 712,36:IF PEEK(53279)=6 THEN 2530
2522 GOTO 2521
2530 GOSUB 90:OPEN #1,6,0,"D1:x.x"
2540 ? CHR$(125);"    DRIVE #1":?
2550 TRAP 2600:FOR J=1 TO 50:INPUT #1,NAME$
2560 ? NAME$:GOSUB 3000+J:NEXT J:GOTO 2600
2600 POKE 712,36:CLOSE #1:? :? "Files loaded.........."
2610 ? "Insert files disk,type disk No.";
2612 TRAP 2615:INPUT D:GOSUB 90:GOTO 2620
2615 CLOSE #1:? F$:? CHR$(253);"You already have a file 
with that     ":? "number,Please type an alternative!    "
2616 GOTO 2612
2620 FILE$(12)=STR$(D):OPEN #1,8,0,FILE$:CLOSE #1
2630 OPEN #1,9,0,FILE$
2640 FOR J=1 TO 50
2650 GOSUB 3050+J:IF NAME$="" THEN 2680
2660 ? #1,NAME$
2670 NEXT J
2680 CLOSE #1:XIO 35,#1,0,0,FILE$:GOSUB 90
2690 ? F$:? "     xxxxxxxFile createdxxxxxxx":FOR T=1 TO
 500:NEXT T:GOTO 10
2700 ? F$:? "  Please type filenames and/or":? :? "
      hit RETURN "
2710 FOR P=1 TO 50:TEMP$="                    ":INPUT A$:IF
 A$="" THEN GOSUB 90:GOTO 2780
2720 FOR I=1 TO LEN(A$)
2730 IF A$(I,I)="." THEN A$(I,I)=" ":IF A$(LEN(A$),LEN(A
$))<>" " THEN 2750
2740 TEMP$(I+2,I+2)=A$(I,I):NEXT I:GOTO 2770
2750 A=11:FOR T=I TO A:TEMP$(T+2,T+2)=" ":NEXT T
2760 FOR J=I+1 TO LEN(A$):TEMP$(A,A)=A$(J,J):A=A+1:NEXT J
2770 NAME$=TEMP$:GOSUB 3000+P:NEXT P
2780 ? F$:? "Please allocate a number to this file."
2790 TRAP 2780:INPUT D:FILE$(12)=STR$(D)
2800 TRAP 2870:OPEN #1,8,0,FILE$:CLOSE #1
2810 OPEN #1,9,0,FILE$
2820 FOR J=1 TO 50:GOSUB 3050+J:IF NAME$="" THEN 2850
2830 ? #1,NAME$
2840 NEXT J
2850 CLOSE #1:XIO 35,#1,0,0,FILE$:GOSUB 90
2855 ? F$:? "     xxxxxxxFile createdxxxxxxx":FOR T=1 TO
 500:NEXT T:GOTO 10
2870 CLOSE #1:? F$:? CHR$(253);"You already have a file 
with that     ":? "number,Please type an alternative!    "
2875 GOTO 2780
```

```
3001 A1$=NAME$:RETURN
3002 A2$=NAME$:RETURN
3003 A3$=NAME$:RETURN
3004 A4$=NAME$:RETURN
3005 A5$=NAME$:RETURN
3006 A6$=NAME$:RETURN
3007 A7$=NAME$:RETURN
3008 A8$=NAME$:RETURN
3009 A9$=NAME$:RETURN
3010 A10$=NAME$:RETURN
3011 A11$=NAME$:RETURN
3012 A12$=NAME$:RETURN
3013 A13$=NAME$:RETURN
3014 A14$=NAME$:RETURN
3015 A15$=NAME$:RETURN
3016 A16$=NAME$:RETURN
3017 A17$=NAME$:RETURN
3018 A18$=NAME$:RETURN
3019 A19$=NAME$:RETURN
3020 A20$=NAME$:RETURN
3021 A21$=NAME$:RETURN
3022 A22$=NAME$:RETURN
3023 A23$=NAME$:RETURN
3024 A24$=NAME$:RETURN
3025 A25$=NAME$:RETURN
3026 A26$=NAME$:RETURN
3027 A27$=NAME$:RETURN
3028 A28$=NAME$:RETURN
3029 A29$=NAME$:RETURN
3030 A30$=NAME$:RETURN
3031 A31$=NAME$:RETURN
3032 A32$=NAME$:RETURN
3033 A33$=NAME$:RETURN
3034 A34$=NAME$:RETURN
3035 A35$=NAME$:RETURN
3036 A36$=NAME$:RETURN
3037 A37$=NAME$:RETURN
3038 A38$=NAME$:RETURN
3039 A39$=NAME$:RETURN
3040 A40$=NAME$:RETURN
3041 A41$=NAME$:RETURN
3042 A42$=NAME$:RETURN
3043 A43$=NAME$:RETURN
3044 A44$=NAME$:RETURN
3045 A45$=NAME$:RETURN
3046 A46$=NAME$:RETURN
3047 A47$=NAME$:RETURN
3048 A48$=NAME$:RETURN
3049 A49$=NAME$:RETURN
3050 A50$=NAME$:RETURN
```

```
3051 NAME$=A1$:RETURN
3052 NAME$=A2$:RETURN
3053 NAME$=A3$:RETURN
3054 NAME$=A4$:RETURN
3055 NAME$=A5$:RETURN
3056 NAME$=A6$:RETURN
3057 NAME$=A7$:RETURN
3058 NAME$=A8$:RETURN
3059 NAME$=A9$:RETURN
3060 NAME$=A10$:RETURN
3061 NAME$=A11$:RETURN
3062 NAME$=A12$:RETURN
3063 NAME$=A13$:RETURN
3064 NAME$=A14$:RETURN
3065 NAME$=A15$:RETURN
3066 NAME$=A16$:RETURN
3067 NAME$=A17$:RETURN
3068 NAME$=A18$:RETURN
3069 NAME$=A19$:RETURN
3070 NAME$=A20$:RETURN
3071 NAME$=A21$:RETURN
3072 NAME$=A22$:RETURN
3073 NAME$=A23$:RETURN
3074 NAME$=A24$:RETURN
3075 NAME$=A25$:RETURN
3076 NAME$=A26$:RETURN
3077 NAME$=A27$:RETURN
3078 NAME$=A28$:RETURN
3079 NAME$=A29$:RETURN
3080 NAME$=A30$:RETURN
3081 NAME$=A31$:RETURN
3082 NAME$=A32$:RETURN
3083 NAME$=A33$:RETURN
3084 NAME$=A34$:RETURN
3085 NAME$=A35$:RETURN
3086 NAME$=A36$:RETURN
3087 NAME$=A37$:RETURN
3088 NAME$=A38$:RETURN
3089 NAME$=A39$:RETURN
3090 NAME$=A40$:RETURN
3091 NAME$=A41$:RETURN
3092 NAME$=A42$:RETURN
3093 NAME$=A43$:RETURN
3094 NAME$=A44$:RETURN
3095 NAME$=A45$:RETURN
3096 NAME$=A46$:RETURN
3097 NAME$=A47$:RETURN
3098 NAME$=A48$:RETURN
3099 NAME$=A49$:RETURN
3100 NAME$=A50$:RETURN
```

MORSE KEYBOARD

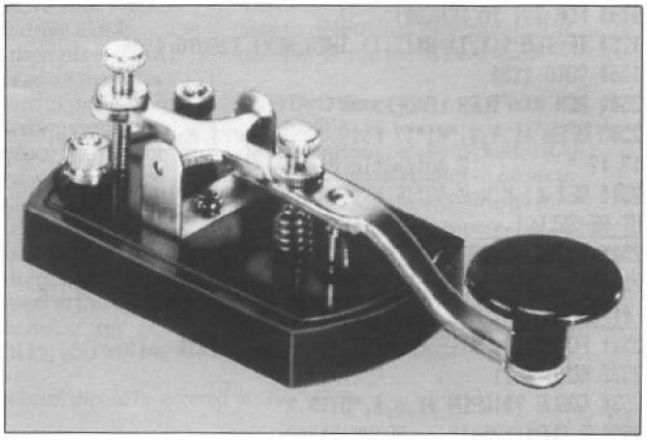
Runs in 16K Cassette or 32K Disk

By Chris Barlow

In an earlier issue we included a program called morse tutor. This program, 'Morse Keyboard', is a follow up which allows you to transmit morse, from pre-typed messages of your own composition, which have been stored on disk, or you can send direct from memory. The program includes the software control of joystick port 1 to control a simple hardware interface to a radio transmitter. See the circuit diagram of the interface. The use of the program is mainly intended for class A radio amateur licence holders. To obtain this licence you must firstly pass the city and guilds examination and the GPO morse examination at 12 words per minute (send/receive). The morse tutor program was written for this purpose. Although you may not have any interest in amateur radio or morse code, the program does contain several interesting routines that deal with disk file management and also contains 'XIO', keyboard 'GET' routines. Also in the program is a small machine code routine to 'disable' the attract mode. This was necessary because when using the keyboard 'GET' routine POKE 77,0 which is normally sufficient when used in a loop within the program it would not function since the program is waiting for a key to be pressed. The machine code resets location seventy-seven to zero every fiftieth of a second irrespective of the program running. If while sending morse data you wish to end transmission prematurely you just press any one of START, SELECT or OPTION keys. The inverse and lower case letters have been suppressed and only valid morse characters are entered into the message.

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

```
0 REM .              MORSE KEYBOARD
1 REM .             By Chris Barlow.
10 CLR :PORTA=54016:PACTL=54018:POKE PACTL,56:POKE PORTA
,1:POKE PACTL,60
20 POKE 82,1:POKE 83,38:GOSUB 6700
25 DIM I$(20),FILE$(20),MFILE$(20),Z$(3000):FILE$="XXXXX
XXX":MFILE$=FILE$
30 GRAPHICS 0:SETCOLOR 2,0,0:SETCOLOR 4,7,0:CLOSE #1:OPE
N #1,4,0,"K:":Q=0:P=1
35 TONE=20:RATE=16:CAL=55:WPM=12:A=77:GOSUB 4102
40 R=0:GRAPHICS 0:SETCOLOR 2,0,0:SETCOLOR 4,7,0:POKE 752
,1
41 PRINT "xxxxxxxxxxxx OUTPUT MODE xxxxxxxxxxxxx":PRINT
45 FOR N=1 TO P:IF N=P THEN 4100
46 IF PEEK(53279)<>7 THEN A=77:GOTO 4102
50 A=ASC(Z$(N,N)):R=R+1:IF R>758 THEN GOSUB 6200
55 IF A=126 THEN NEXT N
60 PRINT CHR$(A);
1040 RESTORE A+4000
4000 READ B
4005 IF B=1 THEN C=TONE:X=1:POKE PORTA,1
4006 IF B=2 THEN C=TONE:X=4:POKE PORTA,1
4007 IF B=3 THEN C=0:X=1:POKE PORTA,0
4010 IF B=4 THEN C=0:X=5:POKE PORTA,0
4020 SOUND 0,C,10,10:FOR T=0 TO X*RATE:NEXT T:IF B=4 THE
N NEXT N
4025 GOTO 4000
4032 DATA 3,3,3,3,4
4046 DATA 1,3,2,3,1,3,2,3,1,3,2,4
4048 DATA 2,3,2,3,2,3,2,3,2,4
4049 DATA 1,3,2,3,2,3,2,3,2,4
4050 DATA 1,3,1,3,2,3,2,3,2,4
4051 DATA 1,3,1,3,1,3,2,3,2,4
4052 DATA 1,3,1,3,1,3,1,3,2,4
4053 DATA 1,3,1,3,1,3,1,3,1,4
4054 DATA 2,3,1,3,1,3,1,3,1,4
4055 DATA 2,3,2,3,1,3,1,3,1,4
4056 DATA 2,3,2,3,2,3,1,3,1,4
4057 DATA 2,3,2,3,2,3,2,3,1,4
4061 DATA 2,3,1,3,1,3,1,3,2,4
4063 DATA 1,3,1,3,2,3,2,3,1,3,1,4
4065 DATA 1,3,2,4
4066 DATA 2,3,1,3,1,3,1,4
4067 DATA 2,3,1,3,2,3,1,4
4068 DATA 2,3,1,3,1,4
```

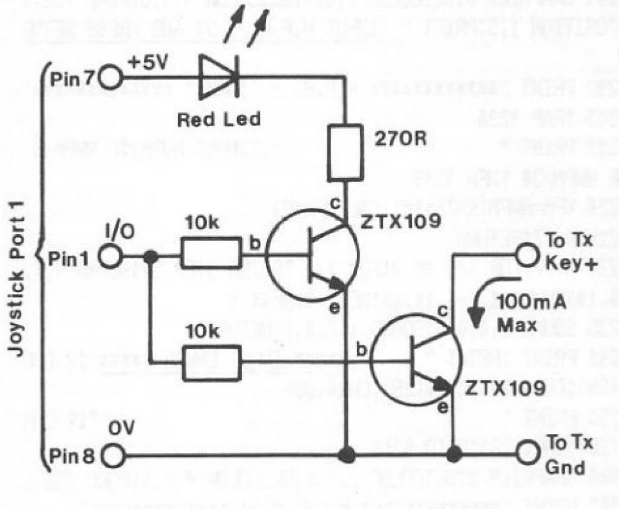


```
4069 DATA 1,4
4070 DATA 1,3,1,3,2,3,1,4
4071 DATA 2,3,2,3,1,4
4072 DATA 1,3,1,3,1,3,1,4
4073 DATA 1,3,1,4
4074 DATA 1,3,2,3,2,3,2,4
4075 DATA 2,3,1,3,2,4
4076 DATA 1,3,2,3,1,3,1,4
4077 DATA 2,3,2,4
4078 DATA 2,3,1,4
4079 DATA 2,3,2,3,2,4
4080 DATA 1,3,2,3,2,3,1,4
4081 DATA 2,3,2,3,1,3,2,4
4082 DATA 1,3,2,3,1,4
4083 DATA 1,3,1,3,1,4
4084 DATA 2,4
4085 DATA 1,3,1,3,2,4
4086 DATA 1,3,1,3,1,3,2,4
4087 DATA 1,3,2,3,2,4
4088 DATA 2,3,1,3,1,3,2,4
4089 DATA 2,3,1,3,2,3,2,4
4090 DATA 2,3,2,3,1,3,1,4
4100 GET #1,A:GOSUB 6600
4102 IF A=79 THEN 40
4103 IF A=84 THEN GOSUB 4400
4104 IF A=83 THEN GOSUB 4200
4105 IF A=73 THEN GOSUB 5900
4106 IF A=68 THEN GOSUB 6300
4107 IF A=70 THEN GOSUB 7025
4109 IF A=77 THEN GOSUB 4300
```

```
4110 GOTO 4100
4200 GRAPHICS 0:SETCOLOR 2,0,0:SETCOLOR 4,4,5:POKE 752,1
:POSITION 1,5:PRINT "  INPUT W.P.M. 5-30 AND PRESS RETUR
N  "
4202 PRINT "xxxxxxxxxxxxx W.P.M. = ";WPM;" xxxxxxxxxxxxx"
4205 TRAP 4230
4210 PRINT "                        ";:INPUT NWPM:IF NWPM<5
OR NWPM>30 THEN 4240
4220 WPM=NWPM:RATE=INT(CAL/WPMx2)
4225 A=77:RETURN
4230 A=77:FOR S=1 TO 3:FOR I=2 TO 255 STEP 50:SOUND 0,I,
10,10:SOUND 1,I-1,10,10:NEXT I:NEXT S
4235 SOUND 0,0,0,0:SOUND 1,0,0,0:RETURN
4240 PRINT :PRINT "        xxxxx INPUT ERROR xxxxx":? CHR
$(28);CHR$(28);CHR$(28);CHR$(28)
4250 PRINT "                                          ":? CHR
$(28);CHR$(28):GOTO 4210
4300 GRAPHICS 0:SETCOLOR 2,0,0:SETCOLOR 4,1,5:POKE 752,1
4302 PRINT "xxxxxxxxxxxxx MORSE MENU xxxxxxxxxxxxx"
4305 PRINT :PRINT "    M=MORSE MENU."
4310 PRINT :PRINT "    I=INPUT MODE."
4320 PRINT :PRINT "  ESC=RETURN TO MORSE MENU."
4330 PRINT :PRINT "    O=OUTPUT MORSE."
4340 PRINT :PRINT "    D=DISPLAY OUTPUT."
4345 PRINT :PRINT "    F=MORSE DATA FILES."
4350 PRINT :PRINT "    S=OUTPUT SPEED,W.P.M. ";WPM;"."
4355 PRINT :PRINT "    T=OUTPUT TONE,5=HI-255=LOW ";TONE
;"."
4360 PRINT :PRINT "    CURRENT MORSE FILE. ";MFILE$;"."
4370 RETURN
4400 GRAPHICS 0:SETCOLOR 2,0,0:SETCOLOR 4,14,8:POKE 752,
1:POSITION 1,5:PRINT " INPUT TONE 5=HI-255=LOW PRESS RET
URN "
4402 PRINT "xxxxxxxxxxxxx TONE = ";TONE;" xxxxxxxxxxxxx
"
4405 TRAP 4420
4410 PRINT "                        ";:INPUT NTONE:IF NTONE<5
 OR NTONE>255 THEN 4430
4415 TONE=NTONE
4417 A=77:RETURN
4420 A=77:FOR S=1 TO 3:FOR I=2 TO 255 STEP 50:SOUND 0,I,
10,10:SOUND 1,I-1,10,10:NEXT I:NEXT S
4425 SOUND 0,0,0,0:SOUND 1,0,0,0:RETURN
4430 PRINT :PRINT "        xxxxx INPUT ERROR xxxxx":? CHR
$(28);CHR$(28);CHR$(28);CHR$(28)
4435 PRINT "                                          ":? CHR
$(28);CHR$(28):GOTO 4410
5900 GRAPHICS 0:SETCOLOR 2,0,0:SETCOLOR 4,3,3
5910 PRINT "xxxxxxxxxxxxx INPUT MODE xxxxxxxxxxxxx"
5920 PRINT "x PRESS E TO ERASE OR A TO ADD INPUT x"
5925 GET #1,A:GOSUB 6600:IF A=69 THEN P=1:Z$="":MFILE$="
XXXXXXXX":GOTO 5990
5930 IF A=65 THEN X=0:GOTO 5937
5935 GOTO 5925
5937 IF P<1 THEN P=1
5940 FOR N=1 TO P:IF N=P THEN GOTO 5960
5945 A=ASC(Z$(N,N)):IF A=126 THEN NEXT N
5950 PRINT CHR$(A);:NEXT N
5960 FOR I=0 TO 255 STEP 25:SOUND 0,I,10,10:NEXT I:SOUND
 0,0,0,0:GOTO 6000
5990 PRINT "xxxxxxxxxxxxxxx READY xxxxxxxxxxxxxxx"
```

```
6000 GET #1,A:GOSUB 6600:IF A=27 THEN 6080
6001 IF A=126 THEN 6050
6002 IF A=47 OR A=58 OR A=59 OR A=60 OR A=62 OR A=64 THE
N 6000
6003 IF A=32 THEN 6010
6004 IF A>90 THEN 6000
6005 IF A<46 THEN 6000
6010 Z$(P,P)=CHR$(A):PRINT CHR$(A);
6020 P=P+1
6022 IF P>2000 AND Q=0 THEN GOSUB 6100
6025 IF P>2999 AND Q=1 THEN GOSUB 6150
6030 GOTO 6000
6050 P=P-1:IF P<1 THEN P=1
6060 Z$(P,P)=CHR$(A):GOSUB 6500
6070 GOTO 6000
6080 A=77:RETURN
6100 PRINT :PRINT "xx WARNING INPUT MEMORY NEARLY FULL x
x"
6110 FOR S=1 TO 5
6120 FOR I=255 TO 5 STEP -20:SOUND 0,I,10,10:SOUND 1,I-2
,10,10:NEXT I:NEXT S
6130 SOUND 0,0,0,0:SOUND 1,0,0,0:Q=1:RETURN
6150 PRINT :PRINT "xxxxxxxxx INPUT MEMORY FULL xxxxxxxxx
"
6160 FOR S=1 TO 5
6170 FOR I=255 TO 5 STEP -20:SOUND 0,I,10,10:SOUND 1,I-2
,10,10:NEXT I:NEXT S
6180 SOUND 0,0,0,0:SOUND 1,0,0,0:Q=1:P=P-1:RETURN
6200 R=0:GRAPHICS 0:SETCOLOR 2,0,0:SETCOLOR 4,7,0
6210 PRINT "xxxxxxxxxxxx OUTPUT MODE xxxxxxxxxxxxx";?
6220 RETURN
6300 R=0:GRAPHICS 0:SETCOLOR 2,0,0:SETCOLOR 4,12,5:POKE
752,1
6310 PRINT "xxxxxxxxx OUTPUT DISPLAY MODE xxxxxxxx";?
6320 FOR N=1 TO P:IF N=P THEN GOTO 4100
6330 A=ASC(Z$(N,N)):R=R+1:IF R>722 THEN GOSUB 6370
6340 IF A=126 THEN NEXT N
6350 PRINT CHR$(A);
6355 IF PEEK(53279)<>7 THEN A=69:GOTO 6395
6360 NEXT N
6370 PRINT :PRINT "xx PRESS C TO CONTINUE OR E TO END xx
":PRINT
6380 GET #1,A:GOSUB 6600
6390 IF A=67 THEN A=32:GOTO 6410
6395 IF A=69 THEN POP :GOTO 4300
6400 GOTO 6380
6410 R=0:GRAPHICS 0:SETCOLOR 2,0,0:SETCOLOR 4,12,5:POKE
752,1
6420 PRINT "xxxxxxxxx OUTPUT DISPLAY MODE xxxxxxxx";?
6430 RETURN
6500 REM DELETE POSITION
6510 IF PEEK(85)>1 THEN ? CHR$(30);" ";CHR$(30);:RETURN
6520 ? CHR$(30);CHR$(28);" ";CHR$(30);CHR$(28);:RETURN
6600 REM CHANGE INVERSE & LOWERS
6610 IF A=155 THEN RETURN
6620 IF A>127 THEN A=A-128
6630 IF A>96 AND A<123 THEN A=A-32:RETURN
6640 RETURN
6700 REM READ M/C
6710 RESTORE 7010
6720 FOR LOOP=0 TO 17:READ D:POKE 1536+LOOP,D:NEXT LOOP
```

```
6730 X=USR(1536):RETURN
7000 REM MACHINE CODE TO RESET ATTRACT
7010 DATA 104,169,6,162,6,160,11,32,92
7020 DATA 228,96,169,0,133,77,76,95,228
7025 GRAPHICS 0:SETCOLOR 2,0,0:SETCOLOR 4,1,5:PRINT "xxx
xxxxx MORSE DATA FILE MENU xxxxxxxx";POKE 752,1
7030 PRINT :PRINT "           D=MORSE FILE DIRECTORY."
7035 PRINT :PRINT "           S=SAVE MORSE FILE."
7040 PRINT :PRINT "           R=RETRIEVE MORSE FILE."
7045 PRINT :PRINT "           K=KILL MORSE DATA FILE."
7050 PRINT :PRINT "           M=RETURN TO MORSE MENU."
7100 GET #1,A:GOSUB 6600
7102 IF A=77 THEN RETURN
7103 IF A=83 THEN GOSUB 8000:GOTO 7025
7104 IF A=82 THEN GOSUB 8500:GOTO 7025
7105 IF A=68 THEN GOSUB 8550:GOTO 7025
7106 IF A=75 THEN GOSUB 8620:GOTO 7025
7110 GOTO 7100
8000 REM WRITE DATA TO DISK
8001 IF LEN(Z$)<1 THEN PRINT :PRINT "    NO MORSE DATA I
N MEMORY TO SAVE    ":GOTO 8003
8002 GOTO 8005
8003 ? CHR$(28);CHR$(28);CHR$(28);:POP :GOTO 7100
8005 GRAPHICS 0:SETCOLOR 2,0,0:SETCOLOR 4,12,5:POKE 752,
1:PRINT "xxxxxxxx WRITE MORSE DATA FILE xxxxxxx"
8010 GOSUB 8200
8060 TRAP 8100:CLOSE #2:OPEN #2,4,0,FILE$
8070 PRINT :PRINT "            FILE ALREADY EXISTS.."
8080 PRINT :PRINT "     DO YOU WISH TO OVER-WRITE (Y/N)"
8081 GET #1,A:GOSUB 6600:IF A=89 THEN 8090
8085 IF A<>89 THEN RETURN
8086 GOTO 8081
8090 XIO 36,#2,0,0,FILE$
8100 TRAP 8700:CLOSE #2:OPEN #2,8,0,FILE$
8110 PRINT #2;Z$
8120 CLOSE #2
8125 MFILE$=FILE$(3,LEN(FILE$)-4)
8130 RETURN
8200 PRINT :PRINT "            ENTER FILE NAME ";:INPUT I$
8205 FILE$=""
8210 FILE$(1,2)="D:"
8213 IF LEN(I$)<1 THEN POP :POP :GOTO 7025
8215 IF LEN(I$)<2 THEN 8230
8220 IF I$(1,2)="D:" THEN FILE$(3,10)=I$(3,LEN(I$)):GOTO
 8240
8230 FILE$(3,11)=I$(1,LEN(I$))
8240 FILE$(LEN(FILE$)+1,LEN(FILE$)+4)=".MRS"
8250 RETURN
8500 REM READ DATA FROM DISK
8505 GRAPHICS 0:SETCOLOR 2,0,0:SETCOLOR 4,12,5:POKE 752,
1:PRINT "xxxxxxxx RETRIEVE MORSE FILE xxxxxxxxx":CLOSE #
2
8510 GOSUB 8200:PRINT
8520 TRAP 8545:OPEN #2,4,0,FILE$
8525 P=1
8530 TRAP 8535:GET #2,X
8531 IF X>90 THEN 8534
8532 Z$(P)=CHR$(X)
8533 PRINT Z$(P);
8534 P=P+1:GOTO 8530
8535 MFILE$=FILE$(3,LEN(FILE$)-4):P=LEN(Z$)+1
```



Here is the electronic circuitry, for you to build, to connect your ATARI to a transmitter.

```
8536 FOR S=10 TO 0 STEP -1:FOR T=50 TO 200 STEP 30:SOUND
 0,T,10,S:SOUND 1,T+1,10,S:NEXT T:NEXT S
8540 RETURN
8545 PRINT :PRINT "            FILE NOT ON DISK......."
8546 PRINT :PRINT "            PRESS ANY KEY TO RETURN"
8547 GET #1,KEY:POP :GOTO 7025
8550 REM READ DIRECTORY
8552 POKE 82,0:POKE 83,40
8555 GRAPHICS 0:SETCOLOR 2,0,0:SETCOLOR 4,12,5:POKE 752,
1:PRINT " xxxxxxxx MORSE FILE DIRECTORY xxxxxxxx":CLOSE
#2
8560 PRINT :FILE$="D:*.MRS"
8570 TRAP 8700:OPEN #2,6,0,FILE$
8580 TRAP 8600:INPUT #2,I$
8583 IF I$(10,16)="SECTORS" THEN ? :? :FOR XY=1 TO 10:?
CHR$(31);:NEXT XY:? I$:GOTO 8580
8585 IF I$(11,13)="MRS" THEN I$=I$(3,10)
8590 PRINT I$,:GOTO 8580
8600 PRINT :PRINT "      PRESS ANY KEY TO RETURN...":POK
E 82,1:POKE 83,38
8610 GET #1,KEY:RETURN
8620 GRAPHICS 0:SETCOLOR 2,0,0:SETCOLOR 4,12,5:POKE 752,
1:PRINT " xxxxxxxx KILL MORSE DATA FILE xxxxxxx":CLOSE #
2
8630 GOSUB 8200
8640 TRAP 8650:OPEN #2,4,0,FILE$
8642 PRINT :PRINT "            (Y/N) TO KILL ";FILE$(3,LEN
(FILE$)-4)
8643 GET #1,A:IF A=89 THEN 8647
8644 IF A<>89 THEN RETURN
8645 GOTO 8643
8647 CLOSE #2:XIO 33,#2,0,0,FILE$:RETURN
8650 PRINT :PRINT "            FILE NOT ON DISK......."
8660 PRINT :PRINT "            PRESS ANY KEY TO RETURN"
8670 GET #1,KEY:POP :GOTO 7025
8700 PRINT :PRINT "             CHECK DISK DRIVE....."
8705 PRINT :PRINT "           PRESS ANY KEY TO RETURN"
8710 GET #1,KEY:POKE 82,1:POKE 38,38:RETURN
```

DOGFIGHT

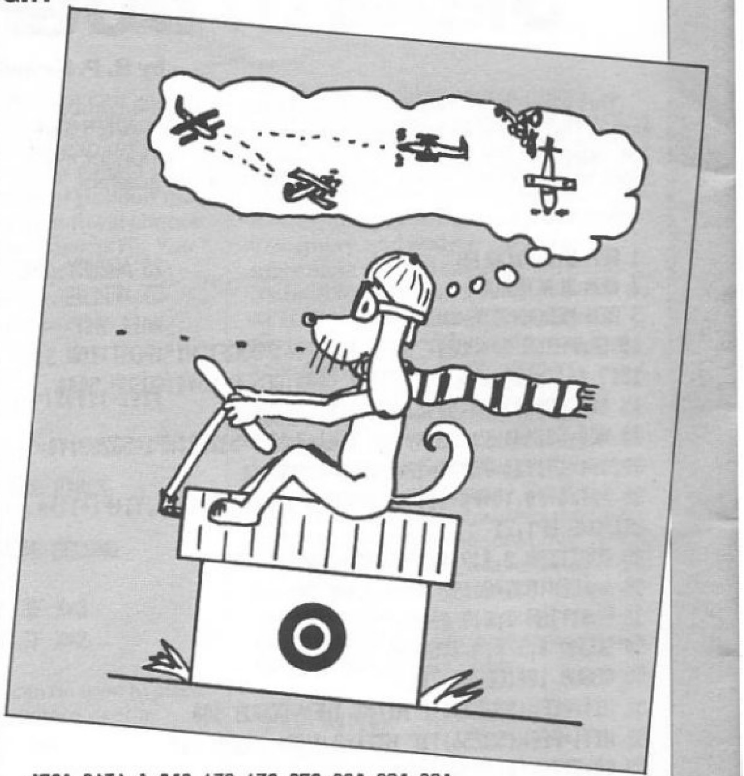
Runs in 32K Cassette or Disk

by R. P. Knowles — St. Helens

This game is for two players, and is an aeroplane combat game. The bi-planes are controlled by joysticks in ports 1 and 2. The object of the game is to shoot down your opponent five times to win. Good use is made of player missile graphics, including the multilayer effect, where the planes disappear behind clouds and buildings, giving you the opportunity to surprise your opponent. The pleasure to be derived from this program makes it well worth the time spent typing it in.

```
1 REM R.P.KNOWLES                        26 ARBURY AVE
2 REM BLACKBROOK                         ST.HELENS
3 REM MERSEYSIDE                         WA11 9HW
10 GRAPHICS 17:POSITION 5,10:? #6;"DOGFIGHT":POSITION 3,
12:? #6;"please wait":GOSUB 4000:GOSUB 4500:GOSUB 5000
15 A$=CHR$(0):A$(4096)=CHR$(0):A$(2)=A$
20 HP0=53248:HP1=53249:M0=0:M1=0:MP0=53252:MP1=53253:Y0=
192:Y1=191:Z0=70:Z1=180:SCR0=0:SCR1=0
30 A$(Y0+P0,Y0+P0+11)=G$:POKE HP0,Z0:A$(Y1+P1,Y1+P1+11)=
B$:POKE HP1,Z1
32 POSITION 2,0:? #6;"PRESS START"
34 A=PEEK(53279):IF A<>6 THEN 34
36 POSITION 2,0:? #6;"           "
40 SOUND 0,170,4,2:SOUND 1,175,4,2
50 GOSUB 100:GOSUB 200
51 HIT=PEEK(53257):IF HIT=3 THEN GOSUB 500
52 HIT1=PEEK(53256):IF HIT1=3 THEN GOSUB 600
55 S0=STICK(0):POKE 77,0
56 IF STRIG(0)=0 AND M0=0 AND S0<>15 THEN DIR0=S0-4:M0=1
:YM0=Y0+1:XM0=Z0+2:POKE MP0,XM0:A$(YM0+M,YM0+M+10)=M$:GO
SUB 150
57 IF M0=1 THEN GOSUB 250
60 S1=STICK(1)
62 IF STRIG(1)=0 AND M1=0 AND S1<>15 THEN DIR1=S1-4:M1=1
:YM1=Y1+1:XM1=Z1+2:POKE MP1,XM1:A$(YM1+M,YM1+M+10)=O$:GO
SUB 150
64 IF M1=1 THEN GOSUB 300
70 GOTO 50
99 REM PLAYER 0 MOVE
100 GOTO STICK(0)+100
105 Y0=Y0+1:Z0=Z0+1:A$(Y0+P0,Y0+P0+11)=E$:GOTO 120
106 Y0=Y0-(Y0>30):Z0=Z0+1:A$(Y0+P0,Y0+P0+11)=D$:GOTO 120
107 Z0=Z0+1:A$(Y0+P0,Y0+P0+11)=G$:GOTO 120
109 Y0=Y0+1:Z0=Z0-1:A$(Y0+P0,Y0+P0+11)=F$:GOTO 120
110 Y0=Y0-(Y0>30):Z0=Z0-1:A$(Y0+P0,Y0+P0+11)=C$:GOTO 120
111 Z0=Z0-1:A$(Y0+P0,Y0+P0+11)=B$:GOTO 120
113 Y0=Y0+2:A$(Y0+P0,Y0+P0+11)=H$:GOTO 120
114 Y0=Y0-2*(Y0>30):A$(Y0+P0,Y0+P0+11)=I$:GOTO 120
115 RETURN
120 POKE HP0,Z0:SOUND 0,85,6,3:IF Z0<56 OR Z0>192 THEN G
OSUB 500
125 IF Y0>192 THEN Y0=192:GOSUB 500
130 RETURN
150 FOR D=0 TO 15:SOUND 2,200+2*D,0,15-D:NEXT D:SOUND 2,
0,0,0:RETURN
199 REM PLAYER 1 MOVE
200 GOTO STICK(1)+200
205 Y1=Y1+1:Z1=Z1+1:A$(Y1+P1,Y1+P1+11)=E$:GOTO 220
206 Y1=Y1-(Y1>30):Z1=Z1+1:A$(Y1+P1,Y1+P1+11)=D$:GOTO 220
207 Z1=Z1+1:A$(Y1+P1,Y1+P1+11)=G$:GOTO 220
209 Y1=Y1+1:Z1=Z1-1:A$(Y1+P1,Y1+P1+11)=F$:GOTO 220
210 Y1=Y1-(Y1>30):Z1=Z1-1:A$(Y1+P1,Y1+P1+11)=C$:GOTO 220
211 Z1=Z1-1:A$(Y1+P1,Y1+P1+11)=B$:GOTO 220
213 Y1=Y1+2:A$(Y1+P1,Y1+P1+11)=H$:GOTO 220
214 Y1=Y1-2*(Y1>30):A$(Y1+P1,Y1+P1+11)=I$:GOTO 220
215 RETURN
220 POKE HP1,Z1:SOUND 1,90,6,3:IF Z1<56 OR Z1>192 THEN G
OSUB 600
225 IF Y1>192 THEN Y1=192:GOSUB 600
230 RETURN
249 REM MISSILE 0 MOVE
250 GOTO 250+DIR0
251 XM0=XM0+4:YM0=YM0+4:GOTO 265
252 XM0=XM0+4:YM0=YM0-4:GOTO 265
253 XM0=XM0+4:GOTO 265
255 XM0=XM0-4:YM0=YM0+4:GOTO 265
256 XM0=XM0-4:YM0=YM0-4:GOTO 265
257 XM0=XM0-4:GOTO 265
259 YM0=YM0+4:GOTO 265
260 YM0=YM0-4:GOTO 265
265 POKE MP0,XM0:A$(YM0+M,YM0+M+10)=M$
270 IF XM0<50 OR XM0>200 OR YM0>200 OR YM0<30 THEN A$(YM
0+M,YM0+M+10)=N$:M0=0
280 RETURN
299 REM MISSILE 1 MOVE
300 GOTO 300+DIR1
301 XM1=XM1+4:YM1=YM1+4:GOTO 315
302 XM1=XM1+4:YM1=YM1-4:GOTO 315
303 XM1=XM1+4:GOTO 315
305 XM1=XM1-4:YM1=YM1+4:GOTO 315
306 XM1=XM1-4:YM1=YM1-4:GOTO 315
307 XM1=XM1-4:GOTO 315
309 YM1=YM1+4:GOTO 315
310 YM1=YM1-4:GOTO 315
315 POKE MP1,XM1:A$(YM1+M,YM1+M+10)=O$
320 IF XM1<50 OR XM1>200 OR YM1>200 OR YM1<30 THEN A$(YM
1+M,YM1+M+10)=N$:M1=0
330 RETURN
499 REM PLAYER 0 FALL ROUTINE
500 A$(YM1+M,YM1+M+10)=N$:M1=0
505 FOR FALL=Y0 TO 190:SOUND 0,FALL,12,12:SOUND 3,FALL-F
ALL/2,10,4:A$(FALL+P0,FALL+P0+11)=H$
510 NEXT FALL:Y0=192:A$(Y0+P0,Y0+P0+11)=J$:FOR D=0 TO 7:
SOUND 0,200,0,15-D:SOUND 3,205,0,15-D:NEXT D
520 A$(Y0+P0,Y0+P0+11)=K$:FOR D=8 TO 15:SOUND 0,200,0,15
-D:SOUND 3,205,0,15-D:NEXT D
530 A$(Y0+P0,Y0+P0+11)=L$:SOUND 0,0,0,0:SOUND 3,0,0,0
540 Y0=192:Z0=70:POKE HP0,Z0:A$(Y0+P0,Y0+P0+11)=G$:SOUND
 0,170,4,2:POKE 53278,0
550 SCR1=SCR1+1:POSITION 19,22:? #6;SCR1:IF SCR1=5 THEN
POP :GOTO 6000
560 RETURN
599 REM PLAYER 1 FALL ROUTINE
600 A$(YM0+M,YM0+M+10)=N$:M0=0
605 FOR FALL=Y1 TO 190:SOUND 1,FALL,12,12:SOUND 3,FALL-F
ALL/2,10,4:A$(FALL+P1,FALL+P1+11)=H$
```

```
610 NEXT FALL:Y1=191:A$(Y1+P1,Y1+P1+11)=J$:FOR D=0 TO 7:
SOUND 1,200,0,15-D:SOUND 3,205,0,15-D:NEXT D
620 A$(Y1+P1,Y1+P1+11)=K$:FOR D=8 TO 15:SOUND 1,200,0,15
-D:SOUND 3,205,0,15-D:NEXT D
630 A$(Y1+P1,Y1+P1+11)=L$:SOUND 1,0,0,0:SOUND 3,0,0,0
640 Y1=191:Z1=180:POKE HP1,Z1:A$(Y1+P1,Y1+P1+11)=B$:SOUN
D 1,175,4,2:POKE 53278,0
650 SCR0=SCR0+1:POSITION 7,22:? #6;SCR0:IF SCR0=5 THEN P
OP :GOTO 6000
660 RETURN
3999 REM SET UP PLAYERS
4000 DIM A$(4096),B$(12),C$(12),D$(12),E$(12),F$(12),G$(
12),H$(12),I$(12),J$(12),K$(12),L$(12),M$(11),N$(11),O$(
11)
4010 FOR X=1 TO 11:FOR Y=1 TO 12:READ Z
4020 IF X=1 THEN B$(Y,Y)=CHR$(Z)
4030 IF X=2 THEN C$(Y,Y)=CHR$(Z)
4040 IF X=3 THEN D$(Y,Y)=CHR$(Z)
4050 IF X=4 THEN E$(Y,Y)=CHR$(Z)
4060 IF X=5 THEN F$(Y,Y)=CHR$(Z)
4070 IF X=6 THEN G$(Y,Y)=CHR$(Z)
4080 IF X=7 THEN H$(Y,Y)=CHR$(Z)
4090 IF X=8 THEN I$(Y,Y)=CHR$(Z)
4100 IF X=9 THEN J$(Y,Y)=CHR$(Z)
4110 IF X=10 THEN K$(Y,Y)=CHR$(Z)
4120 IF X=11 THEN L$(Y,Y)=CHR$(Z)
4130 NEXT Y:NEXT X
4140 DATA 0,0,0,0,248,83,255,248,64,0,0,0
4150 DATA 0,0,32,48,216,236,112,186,31,2,0,0
4160 DATA 0,0,4,12,27,55,14,93,248,64,0,0
4170 DATA 0,0,64,224,72,108,118,59,28,44,0,0
4180 DATA 0,0,2,7,18,54,106,220,56,52,0,0
4190 DATA 0,0,0,0,31,202,255,31,2,0,0,0
4200 DATA 0,0,0,28,28,8,127,127,28,0,0,0
4210 DATA 0,0,0,56,254,254,16,56,56,0,0,0
4220 DATA 0,0,0,0,0,0,28,157,74,255,0,0
4230 DATA 0,0,0,0,0,0,129,66,28,255,0,0
4240 DATA 0,0,0,0,0,0,0,0,0,0,0
4250 FOR Y=1 TO 11:READ Z:M$(Y,Y)=CHR$(Z):NEXT Y
4260 DATA 0,0,0,0,0,1,0,0,0,0,0
4270 FOR Y=1 TO 11:READ Z:N$(Y,Y)=CHR$(Z):NEXT Y
4280 DATA 0,0,0,0,0,0,0,0,0,0,0
4290 FOR Y=1 TO 11:READ Z:O$(Y,Y)=CHR$(Z):NEXT Y
4295 DATA 0,0,0,0,0,4,0,0,0,0,0
4300 A=ADR(A$)
4310 PMBASE=INT(A/2048)*2048+2048
4320 OS=PMBASE-A
4330 POKE 559,62:POKE 54279,PMBASE/256:POKE 53277,3
4340 POKE 704,52:POKE 705,196:POKE 623,4
4350 M=OS+768:P0=OS+1024:P1=OS+1280
4360 RETURN
4499 REM EXCHANGE CHARACTERS
4500 CHBASE=PEEK(106)-6:OLD=57344:NWCH=CHBASE*256
4510 FOR X=0 TO 511:C=PEEK(OLD+X):POKE NWCH+X,C:NEXT X
4520 NWCH=NWCH+8:FOR X=NWCH TO NWCH+87
4530 READ C:POKE X,C:NEXT X
4540 DATA 255,255,255,255,255,255,255,255
4550 DATA 255,153,153,255,255,255,255,255
4560 DATA 7,5,5,7,7,5,5,7
4570 DATA 224,224,224,160,160,224,224,224
4580 DATA 0,31,53,53,63,7,7,7
```



```
4590 DATA 0,248,172,172,252,224,224,224
4600 DATA 95,165,153,153,153,153,165,195
4610 DATA 28,31,63,127,255,255,127,63
4620 DATA 0,140,254,255,255,254,255,254
4630 DATA 63,31,63,127,255,63,15,7
4640 DATA 254,248,252,254,252,248,224,128
4650 POKE 756,CHBASE
4999 REM SET UP PLAYFIELD
5000 POSITION 5,10:? #6;"          ":POSITION 3,12:? #6;"
        "

5005 SETCOLOR 0,1,2:SETCOLOR 1,13,8:SETCOLOR 2,4,4:SETCO
LOR 3,0,12:SETCOLOR 4,7,8
5010 COLOR 1:PLOT 0,21:DRAWTO 19,21:POSITION 0,22:? #6;"
SCORE  0"
5020 POSITION 12,22:? #6;"SCORE  0":COLOR 34:PLOT 8,20:D
RAWTO 11,20
5030 COLOR 35:PLOT 9,19:COLOR 36:PLOT 10,19:COLOR 37:PLO
T 9,18:COLOR 38:PLOT 10,18
5040 COLOR 167:PLOT 0,0:DRAWTO 0,20:PLOT 19,0:DRAWTO 19,
20
5050 COLOR 136:PLOT 9,5:PLOT 8,3:PLOT 10,3:COLOR 137:PLO
T 9,3:PLOT 11,3:PLOT 10,5
5055 COLOR 138:PLOT 8,4:PLOT 9,6:PLOT 10,4:COLOR 139:PLO
T 9,4:PLOT 11,4:PLOT 10,6
5060 RETURN
5999 REM END GAME ROUTINE
6000 SOUND 0,0,0,0:SOUND 1,0,0,0:FOR MUS=1 TO 14:READ NT
6004 POSITION 4,23:? #6;"game over":A$(Y0+P0,Y0+P0+11)=
L$:A$(Y1+P1,Y1+P1+11)=L$
6006 A$(YM0+M,YM0+M+10)=N$:A$(YM1+M,YM1+M+10)=N$
6010 SOUND 0,NT,10,10:FOR D=1 TO 20:NEXT D:SOUND 0,0,0,0
:NEXT MUS
6012 DATA 72,42,42,53,72,42,42,53,72,47,53,42,47,53
6014 RESTORE 6012
6020 POSITION 4,23:? #6;"          ":POSITION 7,22:? #6;"
0":POSITION 19,22:? #6;"0":GOTO 20
```

THE U.K.

ATARI

COMPUTER OWNERS CLUB

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