PRICE £1.00

THE U.K.

ATARI

COMPUTER OWNERS CLUB

INDEPENDENT USER GROUP

ISSUE 5

THE U.K. ATARI COMPUTER OWNERS CLUB

INDEPENDENT USER GROUP

CONTENTS

LIBRARY CHANGES

Our thanks go out to all those who re-subscribed to the club, and we would like to express our sincere appreciation of the knock-out comments we received about the club. Most were complimentary, but there were a few gripes amongst the torrent of good wishes. We appreciate these too, or how else would we know how to improve the club facilities? We also wish to extend a warm welcome to all the new members who are receiving their first newsletter, we hope you like it! Note that copies of issue 4 are still available, price £1 plus 30p postage.

Four suggestions came up time and time again in the comments from members. These were: more on Machine Code, Hardware add-ons and modifications, Software Reviews and last but not least, the inaccessibility of the library to non-programmers. So in this issue you will find major changes to the library rules so that more of the members can take advantage of the system. As to the other suggestions, we have started a series of articles on machine code programming, called 'Cracking the Code'. There is an extensive article on hardware modifications to give improved picture and sound quality, plus a modification to the 400 to give video output for use with a monitor. We were surprised to find that many of you want to see software reviews in the newsletter, we had thought that there were plenty of mags on the book-stalls that cover this aspect of the ATARI, in fact that's about all many of them do cover for the ATARI computers. Still your wish is our command, and in this issue we have a review of the new programming language ACTION!

Finally may we remind you that the newsletter is open to all members to make their own contributions, be it a program, hint, review or anything.

So why not write in, you may have hidden talents!

CREDITS

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

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

BASICALLY SECURE

by Ron, Roy, and Keith.

Have you just written the best program ever created in BASIC? Want to stop your 'best' friends from 'ripping it off'? Don't rush out and buy yourself a shotgun, protect your programs using just a few of the the following techniques! These little programming hints are not guaranteed to protect against the hardened criminal element of the world of software piracy. But, you can stop most people examining and saving off your fantastic programs.

Tip number 1.

This is the simplest, and by far the safest method of protection, known as the LOCKIT & STOW method. This comprises of filing your disks or cassettes into a thief proof, alarm wired container and then padlocking it and putting the key back in the container! As you can imagine this will certainly protect your software from nimble fingers, and as an added bonus it is now impossible for you to accidentally erase your programs.

Tip number 2.

Having realised the above method does have one or two drawbacks, you may wish to dig out that old dusty tool box of yours and spend a few hours with that rusty old hacksaw trying to retrieve your program without damaging it! After ruining your best filing cabinet, make a back up of your program, which you should have done before putting the key inside the box, and try the following.

Tip number 3.

Imagine this is your fantastic, wonderful, exciting BASIC program, it's not, but you have got some imagination haven't you?

```
10 T=0:GOSUB 200
11 T=1:GOSUB 200
12 GOTO 100
200 FOR V=15 TO 0 STEP-1
210 SOUND 0,T,0,V
220 NEXT V
230 RETURN
```

The average life expectancy of your 'BREAK' key is quoted as typically ten million operations. This means that one way of disabling the BREAK key is to operate it ten million and one times! This could wear your fingers to the bone, so here is an alternative method. By adding the following line containing only two 'POKE's (think of the time you'll save!) you can achieve the same result.

```
90 POKE 16,64:POKE 53774,64
```

What happens when the BREAK key is pressed is that the POKEY chip first checks the contents of its internal register 53774. The bits in this register are 'flags' to indicate whether particular interrupts are enabled, or allowed by the user, BREAK is one of these. To disable an interrupt, you need to set the relevant bit in this location to zero, so you could simply disable the BREAK key by POKEing 53774 with zero, thus disabling all interrupts. The only problem with this is that the keyboard interrupt will also be disabled.

The value of 64 is chosen to disable the BREAK key without disabling the keyboard, if your program wishes to utilise any of the other system interrupts, such as count down timers, then you must adjust this POKE value accordingly. The 'POKE 16,64' is used to keep a note of the contents of the POKEY register (53774) which cannot be PEEKed. So if another part of the program needed to know the contents of the register, it can find a copy of it in location 16. If you use this method, you will need to bear in mind that there are many operating system functions used by BASIC which re-enable the BREAK key, so you will have to re-disable it after these. An example is the GRAPHICS command.

Tip number 4.

This method is not very subtle, for it involves changing a warmstart into a coldstart. If you are not sure what this means, then try writing a short program on your ATARI and then press SYSTEM RESET. This is a warmstart, because the system has not cleared the memory, if you LIST you will find that your program is still in memory.

Now POKE 580,1 and repeat the exercise. You should find that when you hit SYSTEM RESET the screen goes blank for a brief moment, just as when the computer is switched on. Now try to LIST the program, and you will find that it has been erased! The POKE 580,1 has forced the machine to follow its power-up procedure when the RESET button is pressed.

Tip number 5.

This tip also involves the SYSTEM RESET key, but is nowhere near as drastic as the last one! It would be far better to obtain some form of control over the RESET button.

Normally, when the RESET button is pressed it causes the machine to go to the start of the BASIC cartridge and print 'READY'. But if location 9 is set to the value of 255 (by 'POKE 9,255') then it jumps to the address found in locations 2 and 3. Normally this is used for a cassette booted program and is supposed to return to the operating system, but if you point to the middle of the BASIC cartridge by POKEing 2,65 and POKEing 3,185 then it will not return, but produce an ERROR instead. Now if you have a TRAP command in your program, then it will jump to the specified line and hence continue your program. Add the following lines to the program and try pressing SYSTEM RESET, (after typing RUN!!!).

```
70 POKE 9,255:POKE 2,65:POKE 3,185
80 TRAP 1000
1000 RUN
```

This will work on most programs, but if other TRAP commands have been implemented then the results are unreliable.

Tip number 6.

Tip 6 is an interesting collection of POKE's which can have the uninitiated scratching their heads vigorously! Type NEW then type the following program into your computer, BUT, you must SAVE it before RUNning it!

```
10 P=155
20 A=1:B=2:C=3
30 FOR I=1 TO 3
40 PRINT I,A,B,C
50 NEXT I
32766 FOR I=PEEK(130)+256*PEEK(131)TO PEEK(132)+256
*PEEK(133):POKE I,155:NEXT I
```

Type RUN, and then type LIST. Notice the mess which the program has degenerated to! What we have done is to POKE the character 155, which is the carriage return key code, directly into every byte in the variable name table. Notice, however, that the program will still RUN properly. This is because the BASIC cartridge only ever needs to reference the actual variable name when a LIST is required, or when a new line is added. Try equating P=42+128, or P=125+128. You must add 128 to the poke because the last character of a numeric variable name must be in inverse video (+128). While in immediate mode, try typing the following line-

```
POKE PEEK(138)+256*PEEK(139)+2,0
```

You should find that no matter what you do, the keyboard simply locks up, and is quite useless. What good is this, we hear you cry, for the system cannot now operate! Well, the secret is that this only happens when BASIC tries to get an extra command or program line from the keyboard, and so will only have effect when the program has STOPPED, BREAK'ed or SYSTEM RESET has been pressed. Type the demo program we gave earlier, and then add the following two lines.

```
32766 FOR I=PEEK(130)+256*PEEK(131)
TO PEEK(132)+256*PEEK(133):POKE I,155:NEXT I
32767 POKE PEEK(138)+256*PEEK(139)+2,0:SAVE "C:"
```

Now type 'GOTO 32766', and you should find that the program will SAVE a copy of itself which can only be used by typing RUN "C:", or RUN "D:filename.ext", and cannot be listed.

Tip number 7.

The final tip uses machine code to achieve the same function as tip number 5, except that it does not rely on jumping 'somewhere' into BASIC, but effectively types 'RUN' followed by a RETURN. Using this method involves a larger program but is guaranteed to work every time, regardless of TRAP and other such commands.

Type 'NEW' in immediate mode and then press return, now type in Listing 1. Remember to SAVE your program before running it, this is always good practice.

Run the Listing 1 program, waiting for it to read in its data first and then press SYSTEM RESET, notice that as in the previous example it re-starts the program, just as if RUN had been typed in. An added bonus with the machine code routine is that it takes the contents of location 1700 decimal and stores this in the interrupt enable registers, so that if 1700 contains 192 then the keyboard will function as normal, and if it contains 64 it will disable the BREAK key, the default in the program is 64.

```
1 REM TYPE 'GOTO 20000' TO INITIALISE THE RESET KEY.
10 ? "Try and stop this program."
20 GOTO 10
20000 POKE 2,0:POKE 3,6
20001 REM POKE 9,3 FOR DISK.
20002 POKE 9,2:REM FOR CASSETTE ONLY.
20003 REM POKE 1700,192 FOR KEYBOARD AND BREAK.
20004 POKE 1700,64:REM DISABLE BREAK.
20010 FOR I=0 TO 104:READ D:POKE 1536+I,D:NEXT I
20020 DATA 162,0,142,101,6,189,26,3
20030 DATA 232,232,232,201,69,208,246,142
20040 DATA 100,6,189,24,3,133,203,169
20050 DATA 105,157,24,3,189,25,3,133
20060 DATA 204,169,6,157,25,3,160,15
20070 DATA 177,203,153,105,6,136,16,248
20080 DATA 169,66,141,109,6,169,6,141
20090 DATA 110,6,173,164,6,133,16,141
20100 DATA 14,210,96,172,101,6,192,3
20110 DATA 240,9,185,102,6,238,101,6
20120 DATA 160,1,96,172,100,6,165,203
20130 DATA 153,24,3,165,204,153,25,3
20140 DATA 169,155,208,236,169,155,82,85
20150 DATA 78
```

Listing 1.

How it works.

On pressing SYSTEM RESET the computer first looks at location 9. Bit 0 indicates that a disk drive or an RS232 box has been 'booted' and bit 1 indicates that a cassette has been booted. If bit 0 has been set then the O.S. will jump to an initialisation routine pointed to by the vector at addresses 12 and 13 decimal. This is used by DOS to re-establish its device name and other miscellaneous functions. However if bit 2 is set then the cassette initialisation routine is called, its vector is at locations 2 and 3 decimal and is normally not used at all. The machine code program is called as a subroutine pointed to by the cassette initialisation vector. This program is patched into the screen editor so that as BASIC goes to fetch a line from the editor our program intercepts this and returns the word 'RUN' followed by a carriage return code of 155 decimal. After the last character has been sent then the program restores the editor to its original vectors. For those interested, the source code is shown in Listing 2.

For those of you who own a disk drive and want an 'AUTORUN.SYS' file to run a BASIC program from disk then type in the source code above and replace the 'RUN' command with 'RUN',QUOTE,'D:PROG.BAS', the 'QUOTE' is used to produce the quote mark for the assembler.

```
0100          *=      $0600
0110 ;Editor patch...
0120 ;By Keith Mayhew.
0130 POKMSK   =       $10
0140 OLDTAB   =       $CB
0150 QUOTE    =       $22
0160 CR       =       $9B
0170 HATABS   =       $031A
0180 MASK     =       1700
0190 IRQEN    =       $D20E
0200          LDX     #$00
0210          STX     COUNT           Set to first character.
0220 FIND     LDA     HATABS,X        Get entry...
0230          INX
0240          INX                     point to next.
0250          INX
0260          CMP     #'E             was last one 'EDITOR'
0270          BNE     FIND            no - go back for next.
0280          STX     PNTR            Save index for later.
0290          LDA     HATABS-2,X      Old table (low).
0300          STA     OLDTAB          Save it.
0310          LDA     #VECTAB&$FF     New table (low).
0320          STA     HATABS-2,X      Replace it.
0330          LDA     HATABS-1,X      Same for high byte...
0340          STA     OLDTAB+1
0350          LDA     #VECTAB/256
0360          STA     HATABS-1,X
0370          LDY     #$0F            Point to end of table.
0380 COPY     LDA     (OLDTAB),Y      Get old table.
0390          STA     VECTAB,Y        and copy it.
0400          DEY                     next element.
0410          BPL     COPY            last one, no - go back.
0420          LDA     #NEWVEC-1&$FF   New 'GET' vector.
0430          STA     VECTAB+4        Replace in new table.
0440          LDA     #NEWVEC-1/256   Same for high byte.
0450          STA     VECTAB+5
0460          LDA     MASK            Load interrupt mask.
0470          STA     POKMSK          Store it.
0480          STA     IRQEN           In hardware too.
0490          RTS                     Return to O.S.
0500 ;This is the new 'GET' handler.
0510 NEWVEC   LDY     COUNT           Character pointer.
0520          CPY     #LEN            Last one...
0530          BEQ     EXIT            Yes, do exit routine.
0540          LDA     STRING,Y        Load character in 'A'.
0550          INC     COUNT           Point to next one.
0560 RETURN   LDY     #$01            Set status to good.
0570          RTS                     Return to O.S.
0580 EXIT     LDY     PNTR            Get pointer into 'HATABS'.
0590          LDA     OLDTAB          Point back
0600          STA     HATABS-2,Y      to old table.
0610          LDA     OLDTAB+1        Same for high byte.
0620          STA     HATABS-1,Y
0630          LDA     #CR             Load 'carriage return'
0640          BNE     RETURN          in 'A' and return
0650 PNTR     *=      *+1
0660 COUNT    *=      *+1
0670 STRING   .BYTE   "RUN"
0680 LEN      =       *-STRING
0690 VECTAB   *=      *+16
```

Listing 2.

CRACKING THE CODE Part 1

By Keith Mayhew and Roy Smith

Whenever you write a program in a high-level language, like BASIC or FORTH, before it can be executed or understood by the microprocessor, it must first be converted into machine code. These high-level languages cannot run as fast as machine code because of the extra time consumed in 'interpreting' each line of your program into machine code. An interpreter only converts one line at a time, so the constant flow of the program is being interrupted for conversion of the next line. A more time efficient method of executing a high-level language is to 'compile' the whole program into one large block of machine code, thus there is no time spent converting lines during program execution. This seems a far better way of implementing high-level languages, but there is a problem in that it consumes vast amounts of memory, which is often impractical on a home micro, but it can result in anything from ten to twenty times the speed of the interpreter.

If you could write programs in machine code, then you would, in effect, be getting right to the heart of the machine. This has its advantages and disadvantages. The advantages include faster implementation of your programs, allows you to access all the functions available in your machine, programs take up less memory and are more efficient. The disadvantages are that writing programs in machine code will often take longer, this is because the high-level languages (BASIC etc.) offer an 'abbreviated' command to implement many machine code subroutines which you would have to write yourself when using machine code.

In this series of articles we hope to not only teach you machine code, but also to show you how and when to use it. In practice, it is usually far more rewarding to combine machine code and high-level languages, mainly BASIC, getting the best features from both. Obviously, many commercially available programs are written purely in machine code, but this has been achieved by many hours of hard work, and if you are making money out of the program it is well worth the effort. For most people's purposes the hybrid method would be more suitable, using small sections of machine code to enhance a BASIC program. The main uses of machine code in this context are speeding up graphics and making animation smoother, for instance fine scrolling. Certain things can only be implemented in machine code, i.e. changing colour many times down the screen (DLI's), or updating clocks or continuously checking for a key depression (VBI's).

Number bases.

Before you can start, you must be familiar with number bases 2 and 16, i.e. binary and hexadecimal, as you will be using them repeatedly. You are already familar with base 10, that's decimal, which is the one we normally use. Base 2 and 16 are very similar. The least significant digit, that is the far right hand one, represents the base number to the power of 0, that is always 1 or 'units', the next digit represents the base number to the power of 1, then 2, etc. For binary (base 2), from the least significant digit onwards the numbers represent: 1, 2, 4, 8, 16, 32, 64, 128, 256 etc. As the base number is 2, the only digits used are '0' and '1', so a binary number of '1101' would be equivalent to, working from the right: (1*1)+(0*2)+(1*4)+(1*8)=13, in decimal. See figure 1 for a worked example.

Hexadecimal, base 16, works on the same principle, i.e. the least significant digit onwards represents: 1, 16, 256, 4096, 65536, etc. Obviously, you can represent very large numbers with only a few digits in hexadecimal compared to binary. As the base is 16, you need to use 16 digits, obviously 0 to 9 are alright but 10, 11, etc, are two digits long. To get over this problem, characters are used to represent these last six digits, so the complete set is:
0 1 2 3 4 5 6 7 8 9 A B C D E F,
where A=10, B=11, C=12, D=13, E=14, F=15. Let's work out an example. To see what '3FB' in hexadecimal represents, we start from the right hand digit with: (B*1)+(F*16)+(3*256) or (11*1)+(15*16)+(3*256)=1019, in decimal. For a worked example see figure 2.

We have shown how to convert from binary to decimal and also hexadecimal (hex) to decimal, we will now show how to do the reverse. To convert a decimal number to binary, for example decimal number 213, you must find the largest number of a power of two that will subtract from 213, in our example it is 128 (256 is too high). Subtracting 128 from 213 leaves 85. This means that you must now write a '1' in the '128' column. You must now work your way through the columns to the least significant digit (64, 32, 16, 8, 4, 2, 1) and you must use every column. The next column is '64' and this will subtract from 85, therefore this column is also a '1', leaving 21. The next column '32' is too large to subtract from 21 so you must write a '0' in this column and move on to the next column '16'. 16 from 21 leaves 5 with a '1' in that column. Column '8' has a '0', column '4' will subtract from 5 and leaves 1, thus you have a '1' in column '4'. Column '2' is a '0' and obviously column '1' is a '1'. See figure 3.

To convert from decimal to hex a similar method is used. For example, to convert 602 decimal to hex you again find the largest column to subtract, which is '256', but you must remember that unlike binary where there is only '0's and '1's, the digits used can be from '0' to 'F'. The way to do this is to find out how many times the column number will subtract from the example number. In our case '256' will subtract from 602 twice (602–512) which leaves 90. This means that in column '256' you write a '2'. The next column is '16' which subtracts from 90 five times (90–80) which leaves 10. Therefore in column '16' you write a '5'. Finally, as the remainder will be less than 16, it can be written into the last column, but remember, if the remainder is a number from 10 to 15 you must write its hex equivalent (A to F). In our example you would write an 'A'. See figure 4. Cont on 6.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	Two to the power of
128	64	32	16	8	4	2	1	= column
0	0	1	0	1	1	0	1	Binary
		32 +		8 + 4 +			1	Decimal

= 45

Figure 1

16^3	16^2	16^1	16^0	16 to the power of
4096	256	16	1	= column
F	A	5	B	Hex
15x4096	10x256	5x16	11x1	Decimal

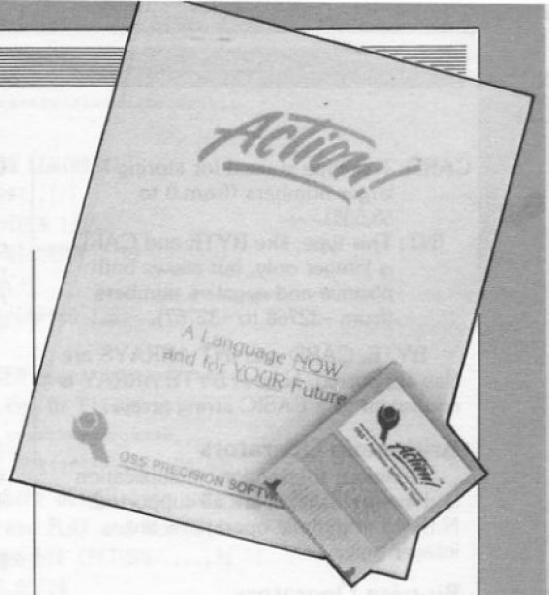
61440 + 2560 + 80 + 11

=64091

Figure 2

ACTION!

By Jon Beff - Manchester



ACTION! is an exciting new language developed by Optimised Systems Software Inc. for the Atari range of computers. The package is supplied in a 24K 'O.S.S. Super-Cartridge', which is accompanied by a 200 page reference manual. ACTION! is really much more than a language, it is a complete programming environment consisting of four distinct parts: a Monitor; an Editor; a Compiler; and a Library of subroutines.

The Monitor is the command centre of the system. It may be used to call the Editor or the Compiler, RUN and SAVE compiled programs, or it may be used to access DOS if a disk-drive is being used. It also includes several useful debugging facilities such as a TRACE function, and a memory dump.

The Editor is where ACTION! programs are written or modified, and as its name would suggest, it is actually a sophisticated text-editor. Its features are too numerous to describe here fully, but amongst others it supports a line length of up to 240 characters; blocks of text can be moved around, and it even allows two text-files to be viewed simultaneously by splitting the display into two text-windows! The Editor is simple to use, and yet it is so powerful it could almost be called a word-processor.

When an ACTION! program has been written it must first be compiled before it can be run. This is the job of the Compiler, which first checks the program for correct syntax, and then if no errors are found it translates the program into machine-code. This process is very fast; small programs are compiled instantaneously, and larger programs can be compiled in a matter of seconds.

The package also contains a Library of around 70 pre-written subroutines which may be called from an ACTION! program. The Graphics, Sound and Joystick functions are all supported; a good variety of Input/Output routines are available, and there are some useful string-handling routines. There are also several miscellaneous routines which do not fit into any of these categories.

One of these, MOVEBLOCK, is a particularly powerful command which allows chunks of memory to be moved around at great speed. Vertical movement of player/missiles, P.M. animation, and page flipping are all easily achieved using MOVEBLOCK.

This brings us to the ACTION! language itself. It was designed to be *the* fastest high-level language available for the Atari, and in this it succeeds admirably. It is hundreds of times faster than BASIC, significantly faster than FORTH, and is only a little slower than programs written in Assembly language.

```
100 GRAPHICS 8+16
110 POKE 19,0: POKE 20,0
120 SCREEN=PEEK(88)+256*PEEK(89)
130 FOR I=SCREEN TO SCREEN+7680
140 POKE I,255
150 NEXT I
160 JIFFIES=PEEK(20)+256*PEEK(19)
170 GRAPHICS 0
180 PRINT "TIME=";JIFFIES;" JIFFIES"
190 END
```

Listing 1

```
PROC SCREENFILL()

    CARD SCREEN, JIFFIES,I
    BYTE TICK=20,
         TOCK=19

    GRAPHICS(8+16)
    POKE(19,0)   POKE(20,0)
    SCREEN=PEEKC (88)

    FOR I=SCREEN TO SCREEN+7680

        DO
        POKE(I,255)
        OD

    JIFFIES=TICK+256*TOCK
    GRAPHICS(0)
    PRINTF("TIME=%U JIFFIES",JIFFIES)

    DO     OD

    RETURN
```

Listing 2

```
PROC SCREENFILL()

    CARD SCREEN, JIFFIES
    BYTE TICK=20,
         TOCK=19

    GRAPHICS(8+16)
    POKE(19,0)  POKE(20,0)
    SCREEN=PEEKC (88)

        SETBLOCK(SCREEN,7680,255)

    JIFFIES=TICK+256*TOCK
    GRAPHICS(0)
    PRINTF("TIME=%U JIFFIES",JIFFIES)

    DO     OD

    RETURN
```

Listing 3

To demonstrate the speed of an ACTION! program I have used a benchmark which fills a Graphics 24 screen with colour (see listings 1, 2 and 3). Listing 1 is a BASIC program which I have included for purposes of comparison. The program itself should be self-explanatory; line 120 finds the start address of screen memory, and lines 130 to 150 form the screenfill loop. The internal real-time clock is used to calculate, in 'Jiffies', the program's execution time. (1 Jiffy = $^1/_{50}$th of a second.) Listing 2 is virtually a direct translation of the BASIC program into ACTION!, and Listing 3 is an ACTION! program which uses the Library routine SETBLOCK to perform the screenfill.

The following results were obtained from running the benchmarks on a 48K Atari 400:

Benchmark Results

	Jiffies	Seconds
Listing 1. (BASIC)	2640	52.80
Listing 2. (ACTION!)	21	0.42
Listing 3. (ACTION!)	3	0.06

The ACTION! translation (listing 2) runs over 120 times faster than its BASIC counterpart (listing 1), and the ACTION! program using SETBLOCK (listing 3) runs 880 times faster than the BASIC program. ACTION! is **FAST!**

ACTION! is a structured language. A structured approach to programming involves controlling the overall design of a program so that its various components fit together neatly. Although this imposes a few restrictions on the programmer, it leads to better thought out programs and more readable code. This is not to say that programming with ACTION! is difficult; it is a delightful language to use. However, the programmer who is used to an unstructured language (such as Atari BASIC) may have to learn a few new techniques.

There follows a summary of the main features of the ACTION! language:

Variable Types

BYTE: This type of variable is used for storing positive integers between 0 and 255.

CARD: This type is used for storing larger numbers (from 0 to 65,535).

INT: This type, like BYTE and CARD, is integer only, but allows both positive and negative numbers (from -32768 to +32767).

BYTE, CARD and INT ARRAYS are also supported. N.B. A BYTE ARRAY is equivalent to a BASIC string array.

Arithmetic Operators

Addition, subtraction, multiplication and integer division are all supported. N.B. All arithmetic operations are integer only.

Bit-wise Operators

Bit-wise operators manipulate numbers in their binary form, allowing the logical operators 'AND', 'OR' and 'EXCLUSIVE OR'. This can be a powerful facility for the experienced programmer.

Relational Operators

The following relational operators are allowed:

= tests for equality.
tests for inequality.
> tests for greater than.
>= tests for greater than or equal to.
< tests for less than.
<= tests for less than or equal to.

Conditional Statements

Conditional statements allow an expression to be tested, and depending on the result determine which part of a program is executed next. The following ACTION! statements allow conditioned execution:

IF ELSE ELSIF
WHILE
UNTIL

WHILE and UNTIL are conditional statements which can be used to control loops.

Procedures

Procedures are an integral part of the ACTION! language. A procedure is a group of actions accomplishing a particular task; each time the task is needed one simply calls the relevant procedure.

Obviously in a review such as this it is not possible to cover all aspects of the ACTION! language, but I hope I have given some idea of its speed and power.

If I sound enthusiastic about ACTION! it's because I am! However, a review without expressing any reservations would be a little one sided . . .

It appears there are still a few bugs present in the ACTION! cartridge, the most annoying of which is a screen which flickers when entering text from the keyboard. It is possible to get used to this, but there should be no need to. The system also occasionally crashes for no apparent reason.

The reference manual is well-written, and contains a lot of information, but is a little scant when covering certain areas, such as using machine-code in an ACTION! program, memory management, and data storage. The manual does not teach how to program using ACTION!, it is more a reference work to show the syntax and the capabilities of the system. The language is crying out for a good tutorial text.

Apparently O.S.S. are aware of these problems, and at the moment are working on a ROM update which should be available as a free replacement to all ACTION! owners. They also say they are going to make available a utility disk which solves the problems with memory management, and also includes several other utilites and demonstration programs.

Don't let the above reservations put you off; ACTION! is a fine language. When its problems have been ironed out, O.S.S. will have developed a package which is worth careful consideration by any Atari programmer.

Continued from 4.

CRACKING THE CODE

128	64	32	16	8	4	2	1	
1	1	0	1	0	1	0	1	213 dec

Figure 3

256	16	1	
2	5	A	602 dec

Figure 4

Converting from binary to hex is very simple, as any hex digit can be expressed by four binary digits. So taking our previous example of 11010101 (213 decimal), it can be split into two groups of four digits, thus working from left to right the first block is 1101, which in decimal is 8+4+1 = 13 or 'D' in hex. The second block is 0101, which in decimal is 4+1 = 5 or '5' in hex. Therefore the hex equivalent of 11010101 is 'D5'. See figure 5 for another example.

To convert the other way i.e. from hex to binary is a reversal of this process. For example, '25A' in hex is three groups of four digits in binary. See figure 6 for a breakdown of the code.

Number bases are prefixed by standard notations in machine code terminology. Binary numbers are prefixed by '%' and hexadecimal numbers by '$'. There is no prefix for decimal numbers. That concludes number bases and conversions, you will be using them constantly when writing in machine code, so it is recommended that if you are not familiar with them, you practice on your own examples.

In future issues we will continue this series delving deeper into machine code as we progress.

1111	0110	
8+4+2+1	4+2	Decimal
F	6	Hex

Figure 5

2				5				A				
2				5				10				Decimal
8	4	2	1	8	4	2	1	8	4	2	1	3 groups of 4
0	0	1	0	0	1	0	1	1	0	1	0	Binary

Figure 6

ATARI D.I.Y.

By Chris Barlow and Bob Kirsch

In this article we will describe several modifications to the hardware of the ATARI 400, 800 and 410 program cassette recorder. At this point, we must make it quite clear that no modifications should be attempted unless you feel that your knowledge of electronics is quite good, and your soldering and construction is of a high standard. You should not consider any modifications if your computer is still under manufacturer's or retailer's guarantee, or any maintenance agreements. The reason for this, is because any modifications would almost certainly invalidate the guarantee on any hardware.

Although the ATARI hardware is excellent, it could be even better, and provide more facilities. The improvements are, in the main, to the video and sound produced on your TV and the extras are a cold-start reset. The modifications can be made to both 400 and 800 models, plus on the 400, direct video and sound can be added. The final improvement is to the 410 program cassette recorder, giving it a 'busy' light like the one on the 810 disc drive.

Better Picture.

The improvements to the video provide a cleaner and sharper picture on your TV screen. The quality of picture given by an ATARI can vary quite a lot. This is due mainly to the amount of unwanted signals on the +5 volt power supply line to the modulator.

The amount of RAM fitted can sometimes make the amount of unwanted signals increase, thus degrading the picture quality. All that is necessary is to add extra de-coupling to the power supply. This should significantly reduce the amount of interference and patterning on the screen. The method for fitting the extra de-coupling on the power supply is shown in Fig. 1.

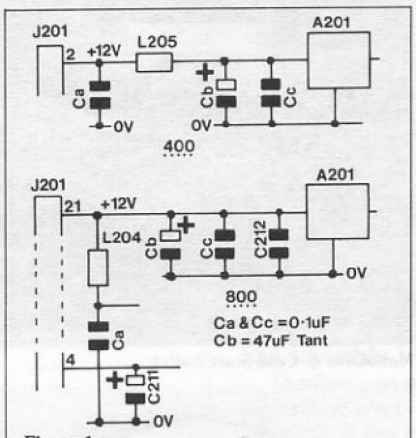
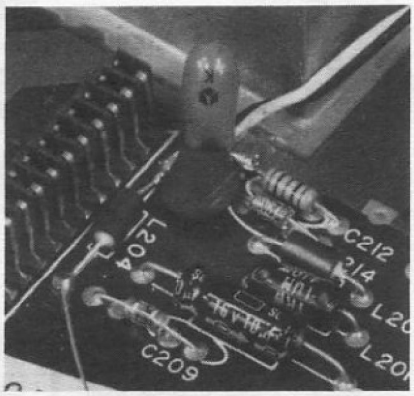


Figure 1.



Decoupling

Following on from this modification, a further improvement is possible on the picture quality and sound reproduction. The fitting of a new, improved modulator gives a sharper picture because of its wide video band width, and clearer sound to your TV speaker. You may have noticed a buzzing sound, even when no sound generators are being used, and the buzzing usually gets worse the more text is on the screen. Sometimes, this is due to the 6 Mhz sound oscillator in the computer being 'off channel'. After fitting, and setting up the new modulator which has its own 6 Mhz sound system, you should notice an improvement in the sound from your TV. When fitting the modulator, it is necessary to disable the ATARI's own 6 Mhz oscillator and feed the sound output to the new modulator (see Fig. 2).

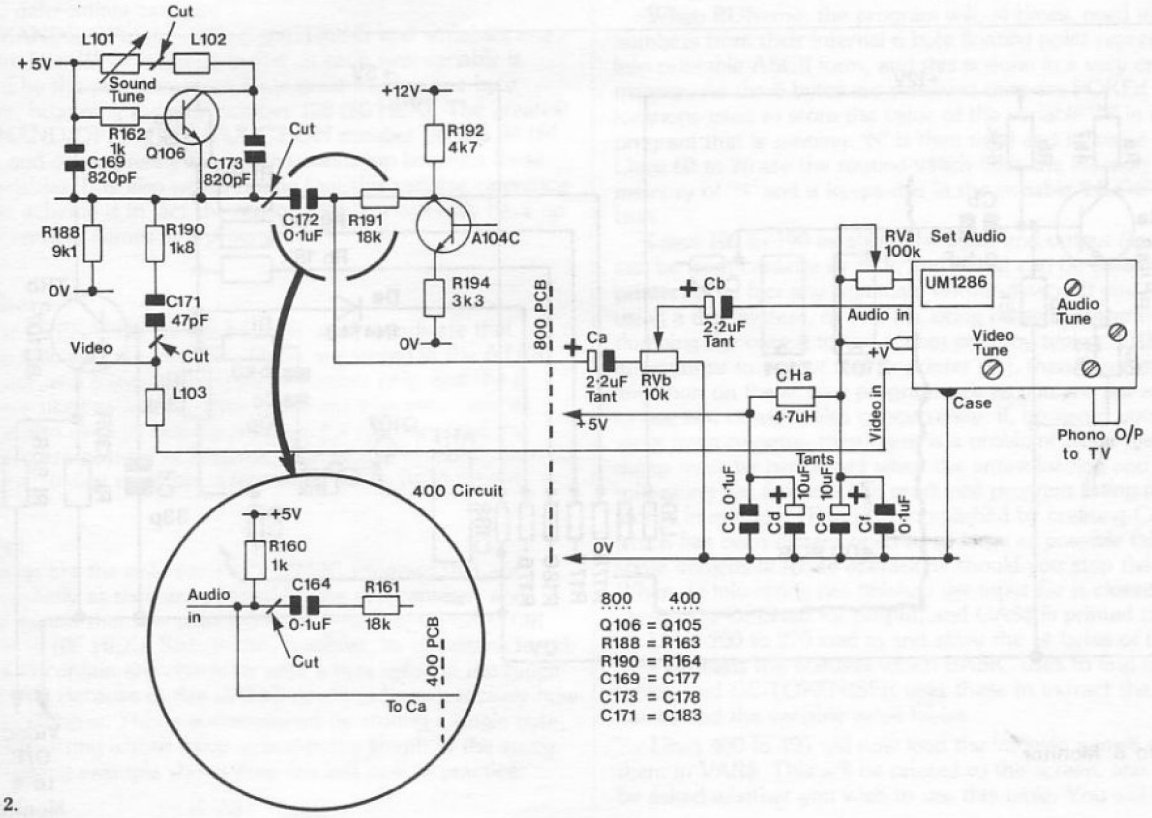
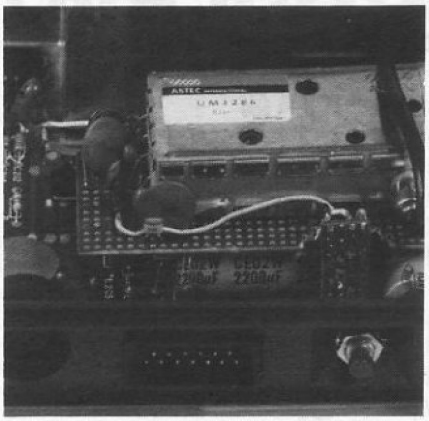


Figure 2.

The new modulator has supply filtering provided by two 10 microfarad tantalum capacitors and a 4.7 microhenry choke. These components help reduce the amount of noise on the supply to the modulator and give a cleaner picture. The method of construction is up to you, but as can be seen in the photos, the new modulator is mounted on a piece of Veroboard fitted in the same place as the original modulator. This was done on an ATARI 800, so the method of mounting the new modulator may be different on a 400. The Veroboard also has the supply filtering components and the pre-sets for sound adjustment, RVa and RVb.

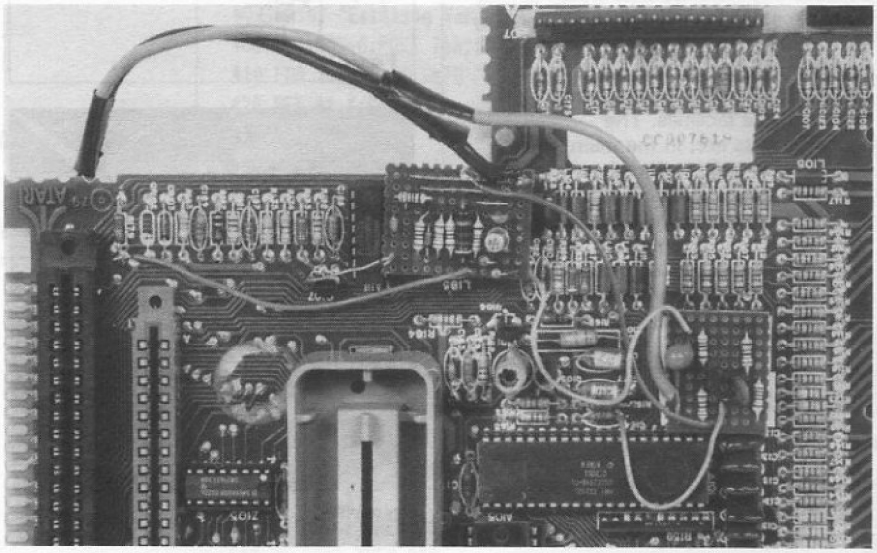
To set up the sound is quite simple. First, set RVa to its halfway setting. Turn up the sound on your TV so that you can hear the buzzing, then adjust with a trimming tool the sound slug on the modulator for minimum buzz and then adjust RVa for fine control. Next, fill the screen with text and, if necessary, readjust RVa for minimum buzz. The other control of the sound is RVb which sets the level of signal fed to the modulator. Too much signal could give rise to distortion on the sound from your TV speaker. Use the sound command 0,100,10,15 for example, to give a loud sound level to feed the modulator. A rough setting of RVb is about halfway, but this control should be set to your own judgement. The other adjustment on the modulator is the TV channel output. If it is necessary to move the channel, this can be done by the video tune slug on the modulator, but this is not usual and in most cases the modulator comes out on a clear channel near to channel 36. Additionally, the new modulator provides a stronger TV signal output giving a more stable picture than the original.



Modulator & Cold Start Switch.

Video 400.

The ATARI 800 has a five pin DIN socket that provides video and audio output for Video Monitors, etc. The ATARI 400 does not have this feature but with a relatively simple modification, both audio and video output can be provided. The circuit shown in Fig. 3 provides audio and composite video – including Chroma. But, if it is intended to feed a black & white or black &



400 Video and Audio mods.

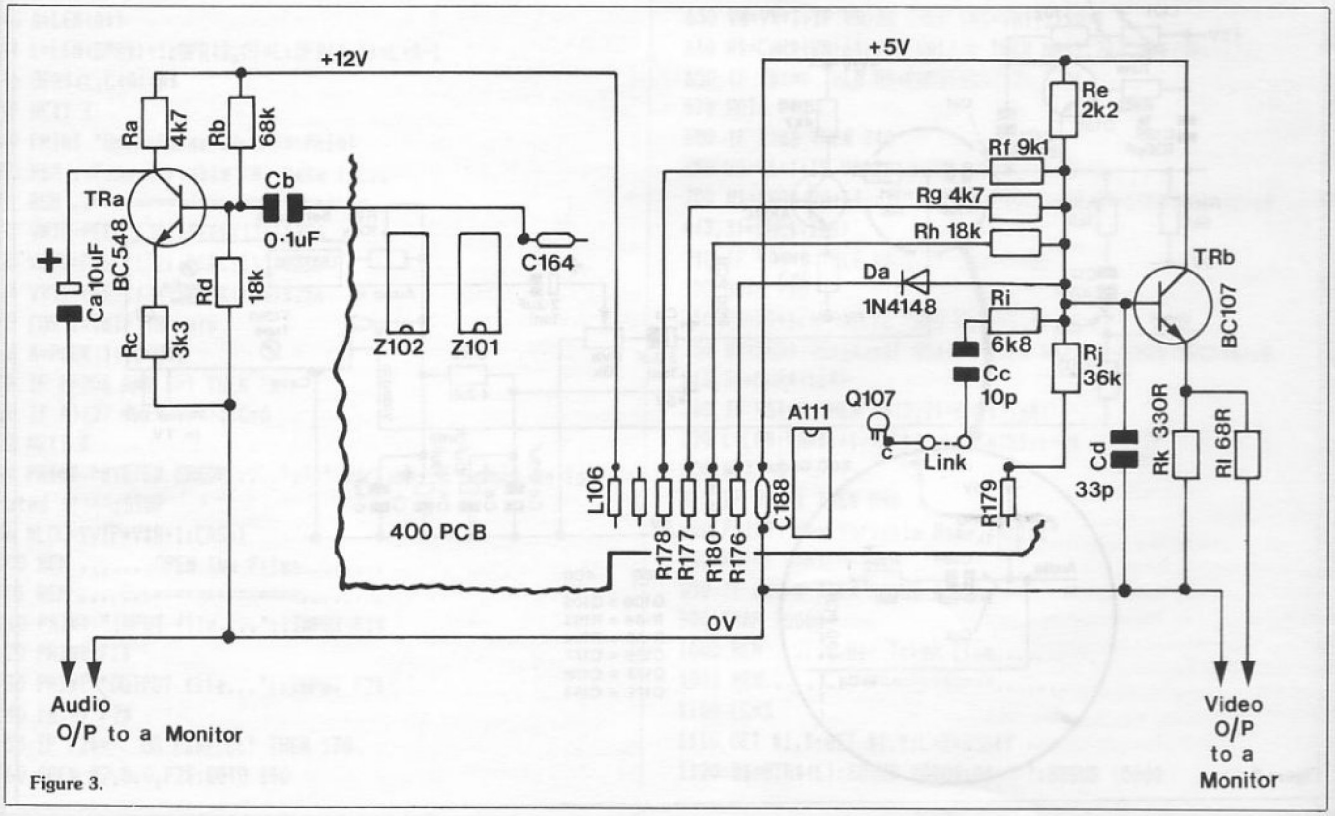


Figure 3.

green type of monitor, the link may be removed in order to prevent colour patterning. A small P.C.B. can be made that provides direct connection to most of the appropriate points, with short wire links, to the remaining connections. The video and audio output leads may be fed out through a small hole drilled in the die-cast case and, in the prototype, we have terminated the leads in a five pin DIN line socket with the same connections as those used in the ATARI 800. An interesting effect will be noted if the connection to R178, is disconnected. This removes the background video information and provides a true white on black display.

Cold Start Key.

When using manufactured software you have probably noticed that the System Reset key does not have the same effect as in your BASIC programs; this is because the manufacturer wants to protect his software. This means that when you press System Reset all that happens is the program restarts or crashes the system, locking up the computer. At this point, all you can do is turn the computer off and on to re-boot another program, which is O.K. if you have a direct video monitor and sound system, but if like most of us, you use a domestic TV, turning off the computer produces a loud, unwanted, hissing sound from the speaker. To have a 'cold start' key on the computer would mean that when it was pushed it would act as an on/off switch and clear the program from memory. If you have a disc drive it will re-boot; or holding down the start key and pressing 'cold start', will prompt the cassette boot load.

The circuit in the ATARI that generates the cold start reset pulse on turn on, can produce further reset pulses at any time using the circuit shown in Fig. 4 and pressing Sa, activating the transistor, momentarily shorting out C198 and producing a new reset pulse within the computer. The fitting of Sa must be done with great care. If it was in a position where it could accidentally be pressed then you would lose your program. For this reason, we decided to fit Sa on the side of the computer near the power on/off switch. On the ATARI 800 there is a convenient position near the power switch, which was used on the American model to select the TV channel for the ATARI. However, in the U.K. model, this switch can be removed and so provide a convenient hole to mount Sa. Unfortunately, the original switch is a slide type and not one of momentory action and not suitable for our purpose. The small amount of electronics involved can be constructed on a small piece of Veroboard mounted on the back of the switch. To use this new facility, simply press Sa and a pulse will be generated, the length of time you press the switch is not important.

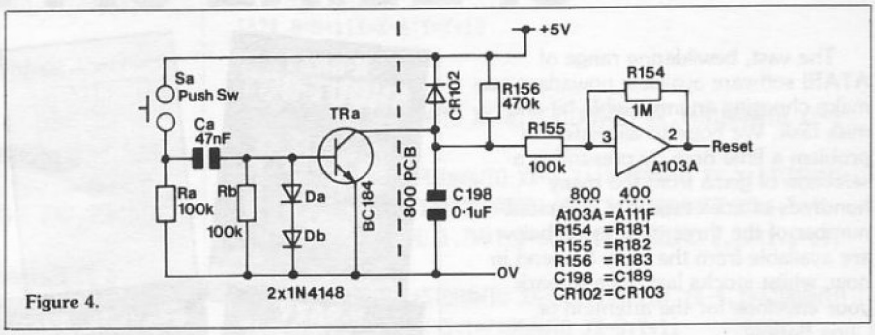
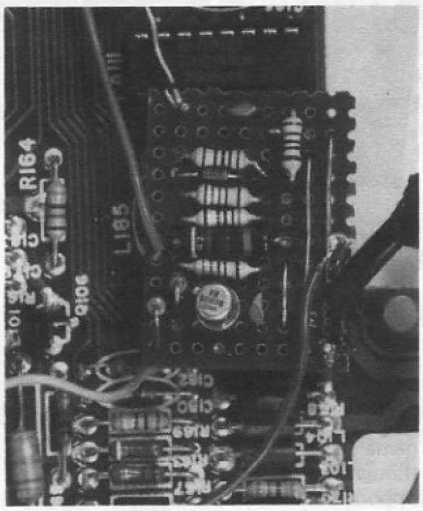
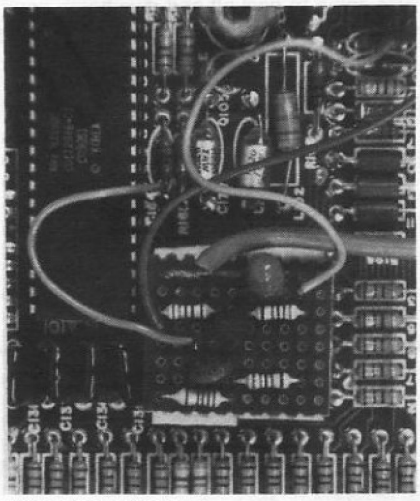
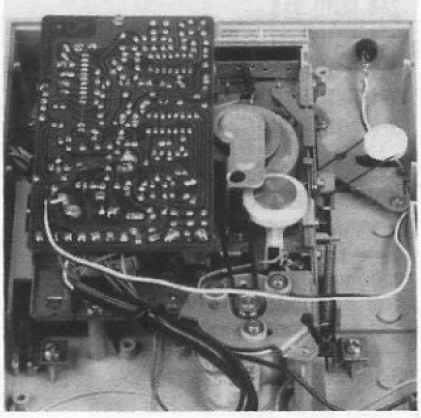


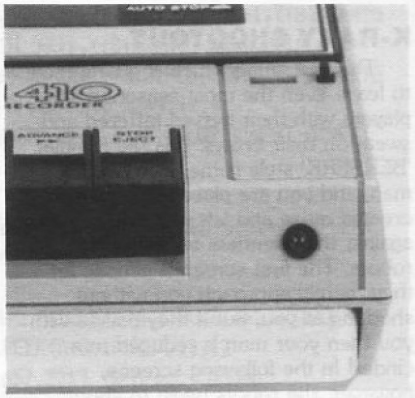
Figure 4.



400 Video modification.


400 Audio modification.


410 Recorder.



Busy Light.

The final modification is to the 410 program cassette recorder and needs hardly any description. When the cassette motor is running, the LED indicator is on, meaning that you don't have to listen to the data going in or out of the recorder, just look to see if the light is on or off. The 1K ohm resistor is to limit the amount of current through the LED. The cassette motor has two wires coming from it, a black and a red,

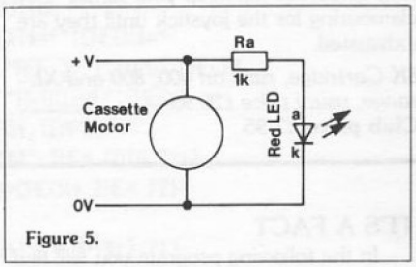


Figure 5.

which connect the motor to the circuit board. The positive supply is on the red wire and the anode of the LED, via the 1K resistor, should be connected to this point. The cathode of the LED is then soldered to the point on the circuit board where the black lead of the motor is connected. Fitting the LED requires the drilling of a hole in the case of the recorder. The hole could be virtually anywhere, but as you can see in the photos, we chose to fit it on the front right hand side of the case. We also used an LED clip to make the mounting a neat job. We find this extra indication very useful.

We hope these modifications will prove to be of some benefit to the more hardware minded of our members, and if any of you have ideas on modifying the ATARI please let us know.

SPECIAL OFFERS

The vast, bewildering range of ATARI software available nowadays can make choosing an impossible hit-and-miss task. We hope to alleviate this problem a little here by presenting a selection of gems from the many hundreds of titles available. A limited number of the three titles listed below are available from the club. So send in now, whilst stocks last. Please mark your envelope for the attention of Chris Barlow.

K-RAZY SHOOTOUT.

This is a game which is guaranteed to leave even the most seasoned game players with their nerves tattered and sweat on their brows. It is the popular 'BERSERK' style game. You are the man, and you are placed in a randomly-created maze and left to defend yourself against the relentless attack of the robots. The first screen is simple, for they merely approach you without shooting at you, but if they collide with you then your man is reduced to a cinder! In the following screens, however, the robots begin to shoot back, and this is where the fun(and the adrenalin) begins. With each screen the robots move faster and shoot sharper, and this is where only lightning-fast reactions can save your men from death. This excellent game is one of our favourites, and it features smooth movements and excellent sound effects, making it one of the most exciting and enthralling games available. Unlike many of the games on the market which look pretty but rapidly become boring, K-razy Shootout will keep your family clamouring for the joystick until they are exhausted.

8K Cartridge, runs on 400, 800 and XL range, usual price £29.95
Club price £7.95

K-RAZY KRITTERS.

Although not as noisy and frantic as K-Razy Shootout, K-Razy Kritters is every bit as addictive. It is based loosely on the 'Space Invaders' theme, you being in control of a base with a formation of invaders above you, but this is where the similarity ends. This game requires stealth and careful thought if you are to outwit your opponents. Not only are you being bombed by the invaders, but they are also going to peel off and dive-bomb you when you shoot them. As if this isn't enough, they actually chase you as they dive, necessitating a well planned run for it by you after you shoot each one. As you lose your men, they are shovelled away by their companions, one of the many novel effects in this excellently produced game. K-Razy Kritters is a thoroughly enjoyable game which I would heartily recommend.

8K Cartridge, 400 & 800 only, usual price £29.95 **Club price £7.95**

GHOST HUNTER.

One of the more popular games in the arcades has been PAC-MAN, a game which Atari have duplicated for use on their home computers, and they have kept to its original format completely. This has meant, however, that there was no variation in the game, and it was not as good as it could have been. Arcade-Plus though, have taken this popular maze theme and created Ghost-Hunter, a far more colourful and interesting version. The graphics are smoother and they have added a little spice to the game by providing seventeen different mazes, and the ability to have two players. Two players can be on the screen alternately, or if you wish, they can be on the screen simultaneously! The usual 'pills' are placed around the maze which your man can eat, thus allowing him to chase and eat the ghosts for a short while! If you are a PAC-MAN fan, then you will find this an excellent and far superior version.

16K Cassette, runs on 400, 800 only, usual price £19.95 **Club price £3.95**

ANALOG MAG.

Certain back issues of the superb ANALOG magazine have become available to members. Issues 11, 12, 13, 14 and 15 are available for £1.80 per copy, which includes postage and packing. There are only limited numbers of each issue, so it's first come first served. ANALOG magazine, which is probably the best of the American ATARI dedicated magazines, usually sells for around £2.30 in the shops, that's if you can find a shop that stocks it! So, if you've missed out on these issues, or you are new to the ATARI scene, don't delay send in today! Don't forget to state which issues you require.

DID YOU KNOWS

IT'S A FACT

In the following program you will find a perfect example of ARTIFACTING, which is a little known (or at least little mentioned) effect that is possible to use. I accidentally stumbled onto this by my experimentations (read playing around) with the Graphics when I first got my machine. It's not a mega-program, but it does show a nice effect when it is demo time!

K. Schrader - Lerwick

```
10 GRAPHICS 8+16:M=0:N=191
20 SETCOLOR 2,8,0:COLOR M
40 PLOT 0,Y:DRAWTO 319,N
54 N=N-1:IF N<0 THEN 200
55 M=M+1:IF M=16 THEN M=0
60 Y=Y+1:GOTO 20
200 X=0:Z=319
210 COLOR M
220 PLOT X,191:DRAWTO Z,0
240 Z=Z-1:X=X+1:M=M-1
250 IF M=0 THEN M=15
255 IF Z<0 THEN 300
260 GOTO 210
300 SETCOLOR 2,B,0
310 B=INT(15*RND(0)+2):FOR W=1 TO 9
99:NEXT W:IF B=5 THEN 310
320 GOTO 300
```

LINE CHECKER

Here is a simple program to help you check programs after they have been typed in. It lists lines one at a time. If an error is found, press BREAK and then type 'L.X' and the current line is listed for you to edit. Once this is done type 'G.10000', not CONT. The reason for this is after the last line has been checked and you are prompted with '?', press BREAK and then CONT and the program will delete itself.

Paul Griffey - Bath.

```
10000 CLOSE #6:CLR :TRAP 10010:DIM
A$(1):X=0:PRINT CHR$(125);"
 START LINE No.":INPUT X:X=X-1:TRAP
 40000
10010 OPEN #6,4,0,"S:":POKE 752,1
10100 X=X+1:? CHR$(125):LIST X
10150 LOCATE 2,2,Z:IF Z=32 THEN 101
00
10170 POSITION 2,2:PUT #6,Z
10200 POSITION 0,22:INPUT A$:GOTO 1
0100
10500 X=10000:? CHR$(125):? :? :? X
:? X+10:? X+100:? X+150:? X+170:? X
+200:? X+500:? X+510:? "POKE842,12"
10510 POSITION 2,0:POKE 842,13:STOP
```

INTERFACE

Printed below are just a small selection of the comments received from members on their re-joining forms.

Great! Do I get a free T-shirt for the shortest comment? Seriously, a problem page and an Adventure help page would be great as well.
Andrew Lusher, Erith.

A good idea, if anybody is having problems or are unable to get past the Ogre in some Adventure game or other, write in and we will see if other members can help.

The best ATARI magazine I have read. I would like to see articles on connecting to the outside world, e.g. to a train set or a simple robot etc.
D. E. Newman, Hayes.

How about some hardware D.I.Y. articles and software & hardware for Morse and RTTY decoding, and of course more and cheaper issues.
R. A. Calkin, Basildon.

Issues 3 & 4 were a great improvement on 1 & 2, keep it up! Lets have some more of the "Did you know . . ." articles, helpful tips make all the difference. Could you do an article on multicoloured players from BASIC?
Jonathan Williamson, Bradford.

We will add your suggestion to our ever growing list of future articles.

Excellent articles on PM graphics and Display Lists, could the same be done for sound effects or DOS? Reviews of software would be useful. Possibly make the newsletters bi-monthly? NO hesitation in renewing membership.
P. D. Blackmore, Harrow.

I like the improved print quality of listings. The Special Offers did not seem very special! although the "slashed prices" are lower in most cases. I have not read issue 4 yet but it looks good!
K. W. Newson, Chelmsford.

I am very impressed with the club's magazine. The presentation, articles and programs are all excellent. I have been able to get more enjoyment from my ATARI thanks to you and the club members.
P. Jenkins, Sheffield.

I find the magazine very interesting, the review of Joysticks was particularly helpful. However, I would like to see less emphasis on the games aspect of the ATARI.
John L. Ball, Swansea.

Great magazine, I especially like the idea of the 3 for 1 software exchange. Could you include more reviews of software and the new ATARI peripherals?
C. Simon, Clwyd.

I have had my 400 48K ATARI now for over 6 months but have not been able to do much programming due to work. But I would like to hire some of the library games, there must be others like myself with little spare time for learning to program at present, but who get great enjoyment playing games.
H.C. Langston, Stockport.

We have had a lot of letters making a similar point. See the Software Library section for some good news.

Keep up the CHR$(199); CHR$(207); CHR$(207); CHR$(196);, CHR$(215); CHR$(207); CHR$(210); CHR$(203).
Michael A. France, Welwyn Garden City.

No software reviews in No.4! This was a disappointment, but balance was O.K. otherwise, and production quality is now fine.Maplin won't sell many Star Raiders, even at £19.95, 99% ATARI owners have already got it I believe!
J. A. Hocking, Tunbridge Wells.

Like issues 3 & 4, well done! Library no good to those with neither time nor talent to write programs.
Jack Schofield, Sutton.

Excellent magazine, especially the utilities like Sector. Worth paying more to keep a nationwide club going.
D. W. Legg, Cardiff.

End of Term Report. Quite good so far, but more software reviews would be appreciated. If required I could submit some of my own! 7 out of 10.
Philip Rae, Hornsey.

The club newsletter is open to all members to submit articles and reviews or anything. We may not always be able to use them but many of the items we receive are used, in fact the newsletter would have quite a few blank pages without your contributions.

Thank you for your excellent magazine, especially the item on adding text to graphics 8. Having little experience of computers I have found your articles very helpful.
D. R. Mills, Holland-on-Sea.

I am getting from the club what I hoped for, a useful magazine and hints & machine code routines beyond my own abilities.
J. A. Spence, Midlothian.

Mag. excellent value, even for my 800 which has three kinds of memory, ROM, RAM and RUM. Frequently I am heard to say "That's RUM!!!" But I keep on trying.
J. L. Chinnery, Rayleigh.

Let's have more hardware articles, perhaps exploring the use of the redundant test point at the back of the 800. I thought issue 3 was great but issue 4 went back to a mag. full of games.
W. Hayes, Tividale.

Keep up the good work, can we have the newsletters a little more frequently? I need 2 machine code subroutines which will FINE scroll 1 mode line in either direction, which can be used as display list interrupts. Can you help?
Grahame Fairall, Shipton-on-Wychwood.

If you would like to write to Keith Mayhew at the club address with details of what precisely you are trying to achieve, he may be able to help.

More info on hardware additions to ATARI 800. More on DOS including Macro Assemblers, printer interfaces and much more on Assembler programs.
Gordon Berry, Kincardineshire.

No constructive comments, just to congratulate you on the improvement of the magazine, and to say that I agree with the selling of the magazine to non-members.
John Hates, Edmonton.

Gets better every time, but don't get too technical!
V. C. Botterill, Hertford.

You are doing a first class job! What about some more information on technical details and secrets!! What is the right hand cartridge slot for on the 800? Need an article on "Assembler Editor". Documentation not very helpful (BUGS!!)
David G. Jones, Normanton.

Nothing is secret on the ATARI, everything you will ever want to know is available in books, such as De Re Atari or Technical Users Notes. Even the circuit diagrams of the computer are obtainable. If you are really stuck on a problem you could always ring the ATARI HELPLINE, Slough (0753) 24561.

Excellent article on display lists, Sector program invaluable. Would like to see more utilities (BASIC renumber, alphabetical sort on a string, etc.) Very readable magazine, I enjoy it very much. Maybe it could be bi-monthly.
Justin M. Baker, Winslow.

I am fluent in BASIC but just started in Assembler/M.C. Information about I/O control in machine code is thin on the ground. Also can you do the equivalent of GR.7 for example without having to write your own display list & allocate screen memory accordingly. Some articles would be very welcome.
D. J. Blease, Weymouth.

One query? What minimum size qualifies a program for inclusion in the software exchange?
Michael Kidd, North Shields.

There is no minimum size as such, but programs donated to the Library should ultimately have a benefit to other members.

Can we have more reviews of software and how about getting together with PAGE 6 and producing one glossy, nationally distributed magazine for ATARI as has been done by a BBC owners club. Please think seriously about it.
Paul Ippaso, Barrow-on-Soar.

Comments Mr. Ellingham??

There seems to be very few books on ATARI compared to say Spectrum. Also there are many game programs one can buy, but not many scientific, mathematical or statistical programs, or am I wrong? Can we be shown how to enter Sinclair programs e.g. "PRINT AT" becomes "PRINT" in ATARI etc.
K. W. Harrison, Glenfield.

Yes you are wrong! There are loads of books available, from the simple BASIC books to the advanced TECHNICAL NOTES. There are serious programs around too, such as VISICALC, CALCULATOR, STATISTICS, MORTGAGE & LOAN ANALYSIS, BOB'S BUSINESS, THE BOOK KEEPER, etc.

An excellent magazine, if not the best ATARI magazine, with superb program listings.
Mary Weightman, Bromham.

Great magazine but how about a check listing as seen in ANALOG and ANTIC to help find typing errors in copying your program listings, PLEASE.
John Stephens, Eastleigh.

An excellent club and magazine only spoilt by the infrequency of issue. Why not make it a monthly or fortnightly issue?
P. Thornber, Bradford.

A great publication which is getting even better, congratulations to those concerned. As programs in the club library are submitted under the banner of "public domain", maybe extra funds could be raised for the club by supplying listings to members for a small fee. Or the same "send a blank tape" as is used, but again with a small fee, for those who cannot write their own.
S. P. Woodward, Isle of Benbecula.

As a person who finds great difficulty in writing programs it seems I will never be able to take advantage of the 3 for 1 exchange. Is there a way I could pay a charge instead?
T. M. Morris, Rhondda.

This newsletter is something that you can all be proud of, better than the I/O magazine from ATARI. Please continue with the excellent work, WE NEED YOU!
J. T. Dunn, Bekesbourne.

I find your major articles to be of great assistance and the quality of program listings equally impressive. However, the program "SPACE FORTRESS" is a direct copy of David Plotkin's work from "Compute's" ATARI GRAPHICS (page 103). I personally bought and read this book some 6 months ago and recognised the program immediately, I am disappointed that your censorship control did not.
P. Hammond, Ushaw Moor.
So are we! So are we!

I would like to see the mag. published once a month, not once in a while!! It's so good! Why not? Quality & contents are marvellous!
Mick Draycott, Ripon.

Could you tell me what pin 9, the Proceed pin, is for on the ATARI 800 I/O port? I can't seem to find any real answer.
Bruce R. Mercer, Worthing.
None of the old peripherals use this pin, but the new range of devices may do. This is an interrupt line which is not supported in the 400/800 operating system, but the vector for that interrupt is supported and the O.S. simply executes a 'PLA' and an 'RTI'. This interrupt can be used in your machine code programs and information can be found in the TECHNICAL NOTES.

In my opinion, if ATARI are going to reach the top of the home micro market, they don't have to drop the price of the machines, but the software prices. I am not clear on your 3 for 1 scheme, if I send in a program on a cassette do I have to enclose another tape for the other copies or do you put them on my tape?
Edward Tilsley, Eastcote.
We agree with you about software prices, but why are you telling us? You should be complaining to ATARI.

How about an article on how to write a war game or make a scrolling map? The game "Dogfight" was excellent and the article on display lists illuminating.
H. Field, Rochester.

I should very much like to see some hints and tips on machine code. I am just starting to understand it and any information would be very useful. I find when using machine code there is a difference to what is written in books and the way it can be used on ATARI. For instance, if you have no disk system fitted can you use the memory that DOS uses for dumping into?
R. Garner, Southport.
Yes.

How about an idiot's guide to DLI's, VBI's, etc. Is it possible to have more than one colour in an individual redefined character? More frequent issues even if we have to pay more.
Peter M. Smith, Grangemouth.
Yes, you can have more than one colour in redefined characters. ANTIC modes 4 & 5 support 4 colours including the background.

My son (age 13) and I read the magazine with great interest, he enjoys typing in the games. I would like to see some articles on interfacing, e.g. analogue input, control of external devices, robotics, etc.
D. G. Moss, Teddington.

May I proffer my thanks to everybody connected with the production of the newsletter. I find it most interesting and have actually managed to learn a few things. Would it be possible to persuade your contributors to include a generous smattering of REMS in the program listings? I realise it is difficult enough to get people to submit programs at all, but it would make the programs more understandable to beginners such as myself. If ever I write anything worthwhile I shall certainly let you have a copy, there are several programs in the library that I would like to have! How about a few listings in Microsoft BASIC with which I am presently struggling?
Derek J. Saxty, Birmingham.

PORT PROBLEMS

Dear Sirs,
Following your article in issue 3, "The ATARI Ports Uncorked", I have been attempting without much success to use the machine to measure voltages produced by outside sources. Could you please tell me if the Paddle inputs only measure resistances placed across pin 7 and pins 5 and 9 or can they measure voltages applied to pins 5 and 9, if so, how? I used pin 8 as the Ground. Also, can pins 1 to 4 on Jacks 1 and 2 (8 lines) be used in conjunction with an external analogue/digital converter, when programmed for input as described in the article. If so, how does one read the input?
B. A. Tenny - Rochdale.
COMMENT
The function of the 8 analogue to digital converters is to measure resistance. It is, however, possible to measure voltage with a small electronic circuit, perhaps using a FET device which would convert your voltage measurement into a resistance which can be read by the ports. Due to the design of POKEY's paddle ports it measures the resistance by charging a capacitor, it should therefore be possible to use the voltage to produce a proportional current which would determine the rate of charging. We have never tried this but in theory it should work.
To read the external device the ports do not need to be reconfigured, and if reading through BASIC then to obtain the eight bit number from jacks 1 & 2 use the following:
VALUE=STICK(1)*16+STICK(Ø)
By the way, in the article in issue 3, a slight error in the diagram showed jacks 1 & 2 incorrectly. Bits DØ to D3 should be jack 1 and bits D4 to D7 should be jack 2.

LIBRARY QUERIES?

Dear Chris,
First, let me congratulate you on your high standard of production. Whilst some people might argue that the newsletter is 'too professional' in appearance the effort put into producing a high quality publication shows that the editorial team are concerned with creating something that will be respected by ATARI users of all levels. Keep up the good work. You have managed to maintain a good mixture of articles covering aspects of the hardware and of various software applications: games, utilities, techniques, etc. Although topics like PM Graphics may have already been the subjects of articles in other publications, another author's presentation of the subject may cast new light on how the technique may be applied. I am particularly pleased that screen displays are now included with the program listings, it is helpful to know what to expect.
Glad to see the growth of the software library. However, I am slightly concerned about some of the library titles. Rule 2 of the library states 'The program you send must be original and not copied'. Unfortunately the following library titles, from their descriptions, are undoubtedly copies of previously published material.
SHOOT:-Originally in COMPUTE! issue 16, Sept. 81, by John H. Palevich.
WEDGE:- Originally in COMPUTE! issue 30, Nov. 82, by Charles Brannon.
KEYBOARD:- Originally in COMPUTE! issue 19, Dec. 81, by Ric Mears.
SPACE FORTRESS:- Originally in COMPUTE! issue 23, Apr. 82, by David Plotkin.
I do not want to cause trouble, but I believe that the original authors should receive their due credit. Incidentally, just because a program is printed in a magazine I understand it does not automatically become 'Public Domain'. I am not sure what you can do about this situation. I would be interested to hear your comments on this matter.
Allen J. Palmer - Basingstoke.
COMMENT.
We are pleased to see that in the main you are happy with the format of the newsletter. We share your concern regarding 'Rule 2' of the library. Although we do try to check every contribution received, it is virtually impossible to identify if a program has been published before. It is obvious to us that the reason people send in programs which are not their own, is to obtain programs when they are unable to write their own. We can tackle the problem in two ways, first the offending programs will be deleted from the library list, and secondly we can change the library rules so that there is no need for people to send in copied programs. If you look at the library pages of this issue you will find that both of these measures have been implemented.

CONTACT

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

SOUTH YORKS
R.A. Horner, 76A Osborne Road, Sheffield, S11 9AZ.

SUFFOLK
Norman Butler Jr. 57 Persimmon Walk, Studland Park, Newmarket, Suffolk. Would like to meet member with Disk Drive in my local area.

CORRIGENDA

ISSUE 4.
Page 11. In the vertical fine scroll program, line 33 should read:
IF PEEK(DL+I)=2 THEN POKE DL+I,2+32

Page 18. Dogfight modification (b), the last statement on line 4520 should read NEXT X instead of NEXT C.

USER GROUP SOFTWARE

Software Librarian - Roy Smith

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

3 FOR 1 EXCHANGE

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

DONATION SCHEME

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

There are numerous reasons why the library rules needed to be changed, some of which have been discussed elsewhere in this issue. But whatever the reasons, I hope that the changes will be for the good and that many more of the members will benefit from them. Finally I would like to point out that some people omit to include return postage when donating to the library, so please do not forget to include 30p worth of stamps to cover this.

THE LIBRARY SOFTWARE SERVICE IS FOR MEMBERS ONLY

LIBRARY SOFTWARE TITLES

Listed below are software titles available to members under the "3 for 1" and donation schemes. As can be seen, they are listed under basic program types, i.e. GAMES, UTILITIES, etc; and also included is the minimum memory required. So if there is a title you fancy, just sent in a program of yours for exchange or donate to the club funds.

Games

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

LIBRARY SOFTWARE

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Adventure Games

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

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

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

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

Home Entertainment

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Demo's

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

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

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

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

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

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

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

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

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

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

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

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

LIBRARY SOFTWARE

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Utilities

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

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

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

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

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

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

CLOCK
by H. Clark - Barking.
Adds a 24 hour clock to the top right-hand corner of the screen, useful for timing games etc.
Runs in 16K Cassette or Disk min.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 min.

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

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

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

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

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

LIBRARY SOFTWARE

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

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

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

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

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

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

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

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

TOP TEN

1 (2)	THE VALLEY	STEVE CALKIN
2 (1)	SYNTHESISER	CHRIS PAYNE
3 (5)	STONEVILLE MANOR	NIGEL HASLOCK
4 (-)	FOLLY OF E. KKHANN	ALEX KELLS
5 (4)	ASSEMBLER	CHRIS RUTTER
6 (7)	LIE DETECTOR	D. DODSON
7 (-)	TUNNEL TRILOGY	MARK CHRISTIAN
8 (9)	OUTPOST	ANTHONY BALL
9 (-)	SEAWARFARE	STEVE TULLETT
10 (8)	ASTRO DODGER	D. DODSON

Well, VALLEY made it to the number one spot after all. Congratulations Steve! The position was reversed this quarter by VALLEY just edging out SYNTHESISER by a few requests. MANOR moved up, but the new rising star must be FOLLY of EZRHAR'D KKHANN, a recent library entry by Alex Kells, which has shot into the chart at number 4. That makes 3 adventure games in the top 4. There must be a lot of midnight adventurers out there. Another interesting new entry is the TUNNEL TRILOGY by Mark Christian, look out for his new library entry, GIL·BERT, which is an excellent game.

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

Education

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

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

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

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

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

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

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

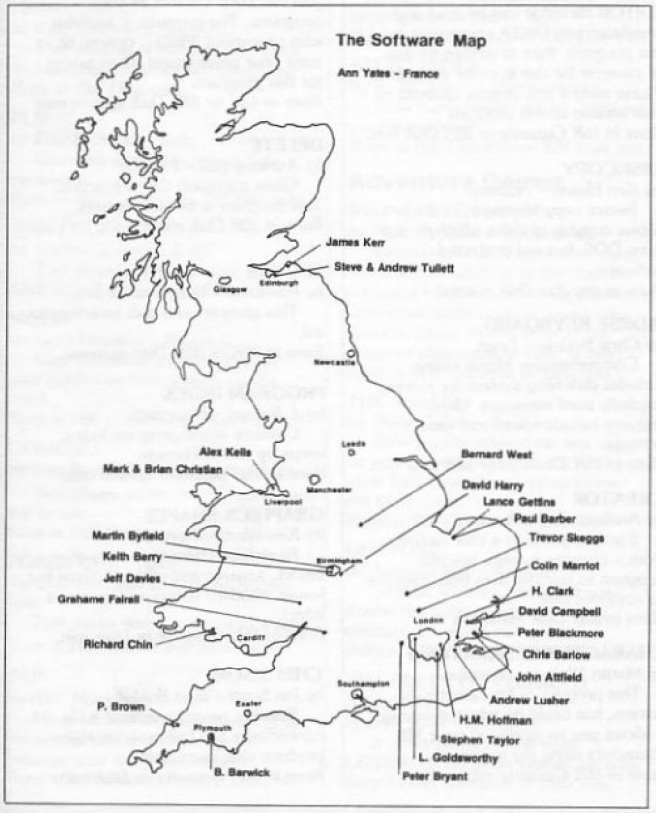
KEYBOARD TUTOR
by Mike Jervis - Nottingham.
Learn to touch type with this typing tutor.
Runs in 16K Cassette or Disk min.

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

Music

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

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



LABEL MAKER

By John Attfield and Peter Blackmore

After many requests for some sort of labelling facility, we have produced the program 'LABEL MAKER', which will print details on computer labels 89mm by 36mm, and will print labels side by side.

The program can be used with either disk or cassette, and because of the novel DATA storage system, it is ideal for the cassette user. The program features on-screen editing which makes it very flexible and very easy for data repetition and therefore very useful, if for example, you run a club or business and often have to send information to a company with several addresses.

There are many options available in the program including Retrieve, Edit, Clear, Print, Disk Save and Cassette Save. Pressing ESC puts you into the command mode. In the edit mode you can input up to 30 characters per line, and there are 7 lines available. This should be adequate for almost all purposes.

This program has been tested on an EPSON RX80 and a MICROLINE 80, and has worked successfully on both.

Control commands are as follows:-

Clear: clears the label edit area. Print: output to printer.
Retrieve: finds in memory a referenced label.
Edit: enter or edit labels.
DISKSAVE: save entire program & new labels to disk.
CASSETTESAVE: as above to cassette. ESC: go to command mode.
LIST: starting with the first label in memory list each in turn.

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

```
25 REM BY J. ATTFIELD & P. BLACKMORE
81 DIM TEXT$(220),TEMP1$(73),TEMP2$(73),TEMP3$(73),CLEAR
$(220),DAT$(300),CHECK$(3):STEP=10
83 REM CLEAR STRING
85 FOR CL=1 TO 210:TEXT$(CL,CL)=" ":NEXT CL:CLEAR$=TEXT$
86 RESTORE 10000:READ LINE,CHECK:GOSUB 750:GOTO 555
110 REM SCREEN DISPLAY
170 POKE 82,2:POKE 83,39:? CHR$(125);:SETCOLOR 2,14,12:S
ETCOLOR 1,0,0:SETCOLOR 4,14,12:LX=36:LY=6:X=5:Y=4
200 ? "                                "
210 ? "  xxxxxxxxx LABEL MAKER xxxxxxxxx"
220 ? "                                "
230 POSITION X,Y:? "|  ESC COMMAND                  |"
240 POSITION X-3,Y+1:? CHR$(17);CHR$(18);CHR$(18);
250 FOR XX=X TO LX-2:? CHR$(23);:NEXT XX:? CHR$(18);CHR$
(18);CHR$(5)
300 FOR YY=6 TO LY+6:POSITION X-3,YY:? CHR$(1);POSITION
LX+1,YY:? CHR$(4):NEXT YY:? CHR$(26);CHR$(18);CHR$(18);
360 FOR XX=X TO LX-2:? CHR$(24);:NEXT XX:? CHR$(18);CHR$
(18);CHR$(3)
400 POKE 82,5:POKE 83,37:? :? "STORE RETRIEVE EDIT CLEAR
 PRINT"
405 ? "DISK SAVE  CASSETTE SAVE   LIST"
406 POSITION 3,20:? "LABEL REFERENCE LAST 3  CHARACTERS"
407 POSITION 3,21:? "YOU  3 LETTERS  COMPUTER 3 NUMBERS"
408 RETURN
430 REM x CHARACTER EDIT ROUTINE x
450 POSITION 21,4:? " x EDIT x "
460 POKE 752,0:POSITION 4,6:? "  ";CHR$(126);
470 GOSUB 690:GET #1,CHA:GOSUB 700
471 IF CHA=27 THEN POKE 752,1:POKE 702,64:RETURN
472 IF CHA=125 THEN POSITION 3,PEEK(84):FOR L=3 TO LX-2:
? " ";:NEXT L:POSITION 4,PEEK(84):? " ";:GOTO 470
475 IF PEEK(85)>34 THEN ? CHR$(155);:GOTO 470
476 IF PEEK(85)<6 THEN IF CHA=30 THEN GOTO 470
477 IF CHA>155 THEN IF CHA<158 THEN 470
478 IF CHA>253 THEN 470
480 ? CHR$(CHA);:GOTO 470
540 REM COMMAND ROUTINE
555 GOSUB 170
570 POSITION 21,4:? "              ":POSITION 20,4:? ">";
590 GOSUB 690:GET #1,CHA:GOSUB 700:IF CHA>128 THEN CHA=C
HA-128
600 IF CHA=83 THEN GOSUB 830:GOSUB 5520:GOSUB 170:GOSUB
3861
610 IF CHA=82 THEN GOSUB 7030:GOSUB 3861
620 IF CHA=69 THEN GOSUB 450:REM EDIT
630 IF CHA=67 THEN GOSUB 1861:REM CLEAR
633 IF CHA=76 THEN GOSUB 8000:REM LIST
640 IF CHA=80 THEN GOSUB 830:GOSUB 6830
641 IF CHA=68 THEN POSITION 21,4:? "SAVE D:LABEL":SAVE "
D:LABEL"
642 IF CHA=65 THEN POSITION 21,4:? "CASSETTE SAVE":CSAVE
650 GOTO 570
690 OPEN #1,4,0,"K:":RETURN :REM OPEN KEYBOARD
700 CLOSE #1:RETURN
750 OPEN #6,12,0,"S:":RETURN :REM OPEN SCREEN
760 CLOSE #6:RETURN
810 REM x      READ SCREEN      x
830 POKE 752,1:POSITION 21,4:? " x READING x ":C=0
840 FOR Y1=6 TO 12:FOR X1=5 TO 34:C=C+1
850 LOCATE X1,Y1,CHAR:POSITION X1,Y1:? CHR$(CHAR+128):IF
 CHAR>128 THEN CHAR=CHAR-128
852 IF CHAR=44 THEN CHAR=32
855 TEXT$(C,C)=CHR$(CHAR):NEXT X1:NEXT Y1:CHAR=0:C=0:RET
URN
1810 REM x     CLEAR SCREEN      x
1861 POKE 752,1
1862 POSITION 21,4:? "x CLEARING x"
1863 TEXT$=CLEAR$:GOSUB 3864:RETURN
3810 REM x  REWRITE TO SCREEN    x
3861 POKE 752,1
3862 POSITION 21,4:? "x WRITING x"
3864 L=0:FOR Y1=6 TO 12:FOR X1=5 TO 34:L=L+1
3865 POSITION X1,Y1:? TEXT$(L,L):NEXT X1:NEXT Y1
3872 POSITION 21,4:? "              "
3880 L=0:RETURN
5510 REM x DATA LINE  PREPARATION x
5520 POSITION 21,4:? "x STORING x"
5530 DAT$="":INC=0:AS=ASC(TEXT$(208,208))
5540 IF TEXT$(208,210)="   " THEN TEXT$(208,210)=STR$(CH
ECK):CHECK=CHECK+1:GOTO 5548
5545 IF TEXT$(208,210)<>"   " THEN IF AS>48 AND AS<58 TH
EN IF VAL(TEXT$(208,210))<=CHECK THEN 6961
5548 IF FRE(0)<500 THEN GOTO 6964
5549 FOR L=1 TO 210 STEP 70:INC=INC+1
5550 DAT$(LEN(DAT$)+1)=STR$(LINE+INC)
5552 DAT$(LEN(DAT$)+1)=" DATA "
5553 DAT$(LEN(DAT$)+1)=CHR$(34)
5560 DAT$(LEN(DAT$)+1)=TEXT$(L,L+69)
5565 DAT$(LEN(DAT$)+1)=CHR$(34)
5570 NEXT L:GOTO 6635
5647 DAT$="":LINE=LINE+STEP
5650 DAT$(LEN(DAT$)+1)="10000"
5652 DAT$(LEN(DAT$)+1)=" DATA "
5660 DAT$(LEN(DAT$)+1)=STR$(LINE)
5661 DAT$(LEN(DAT$)+1)=","
5662 DAT$(LEN(DAT$)+1)=STR$(CHECK)
```

Continued on 19.

GIL-BERT

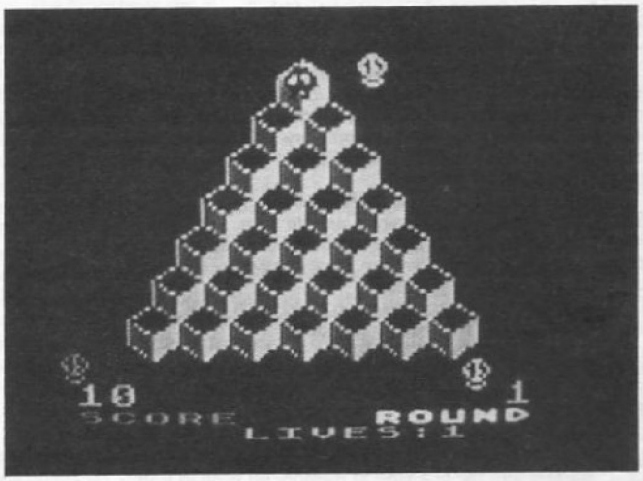
Runs in 32K Cassette or Disk

by Mark & Brian Christian

Bounce Gil-bert up and down the pyramid, changing the colours of the squares. But beware, the SKULLS are chasing you! See how many screens you can clear before they get you. This game uses a joystick and is a great game, similar to SLINKY or PHARAOH'S PYRAMID but written in BASIC for you to type in.

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

```
5 DIM CHAR$(40):PX=1
10 GOSUB 800:GRAPHICS 7+16:POKE 559,46:POKE 53277,3:SCORE=0
:ROUND=1:LIVES=3:POKE 710,0:GOSUB 1000
12 GOSUB 1090
20 GOTO 100
30 FOR O=10 TO 0 STEP -1:SOUND 0,70,12,O:NEXT O
35 CX=XP0-49:CY=YP0+6:COLOR 3:PLOT CX,CY:PLOT CX+1,CY-1:
DRAWTO CX+1,CY+1:PLOT CX+2,CY-1.5:DRAWTO CX+2,CY+1.5
37 PLOT CX+3,CY-2:DRAWTO CX+3,CY+2:PLOT CX+4,CY-2.5:DRAW
TO CX+4,CY+2.5:PLOT CX+5,CY-3:DRAWTO CX+5,CY+3
38 PLOT CX+6,CY-2.5:DRAWTO CX+6,CY+2.5:PLOT CX+7,CY-2:DR
AWTO CX+7,CY+2:PLOT CX+8,CY-1.5:DRAWTO CX+8,CY+1.5
39 PLOT CX+9,CY-1:DRAWTO CX+9,CY+1:PLOT CX+10,CY
40 NUMBLOCK=NUMBLOCK+1:SCORE=SCORE+10
50 IF SCORE/500=INT(SCORE/500) THEN LIVES=LIVES+1
60 IF NUMBLOCK=28 THEN POP :GOTO 500
70 SOUND 0,0,0,0:RETURN
100 ST=PEEK(632):XP0=XP0-6*(ST=10)-6*(ST=9)+6*(ST=5)+6*(
ST=6)
105 YP0=YP0-10*(ST=10)-10*(ST=6)+10*(ST=9)+10*(ST=5):POK
E 77,0
110 IF ST=10 OR ST=6 THEN SHAPE=PM1*(ST=10)+PM2*(ST=6)
114 IF YP0<-2 THEN YP0=-2:GOTO 400
120 POKE 53248,XP0:D=USR(ADR(JUMP$),PM0+YP0,SHAPE):POKE
53278,0
130 FOR N=1 TO 3:RR=PEEK(53770)/ROUND:VEL1=VEL*(XP(N)<XP
0)*(RR<26)-VEL*(XP(N)>XP0)*(RR<26)
140 VEL2=VEL*(YP(N)<YP0)*(RR<26)-VEL*(YP(N)>YP0)*(RR<26)
:XP(N)=XP(N)+VEL1:YP(N)=YP(N)+VEL2
150 D=USR(ADR(JUMP$),PM(N)+YP(N),PM3):POKE 53248+N,XP(N)
:NEXT N
160 IF PEEK(53260)<>0 THEN GOTO 300
170 IF PEEK(53252)=0 OR YP0=70 THEN GOTO 400
180 IF PEEK(53252)<5 THEN GOSUB 30
190 GOTO 100
300 SOUND 0,132,10,4:SOUND 1,132+1,10,4:GOSUB 390:GOSUB
390:SOUND 0,120,10,4:SOUND 1,120+1,10,4:GOSUB 390
302 SOUND 0,112,10,4:SOUND 1,112+1,10,4:GOSUB 390
310 SOUND 0,132,10,4:SOUND 1,132+1,10,4:GOSUB 390:SOUND
0,0,0,0:SOUND 1,0,0,0
320 FOR N=10 TO 100:POKE 704,N:SOUND 0,N,10,4:NEXT N:POK
E 704,N:SOUND 0,0,0,0
330 LIVES=LIVES-1:POKE 704,34
340 D=USR(ADR(JUMP$),PM0+YP0,PMB+100)
350 FOR N=1 TO 3:D=USR(ADR(JUMP$),PM(N)+YP(N),PMB+100):N
EXT N:POKE 53278,1
360 IF LIVES=0 THEN GOTO 600
370 GOSUB 960:GOSUB 1120:GOTO 100
390 L=L+1:SETCOLOR 2,L,14:FOR N=1 TO 100:NEXT N:RETURN
391 L=L+1:SETCOLOR 2,L,14:FOR N=1 TO 50:NEXT N:RETURN
```




```
392 L=L+1:SETCOLOR 2,L,14:FOR N=1 TO 100:NEXT N:RETURN
393 L=L+1:SETCOLOR 2,L,14:FOR N=1 TO 200:NEXT N:RETURN
394 FOR DE=1 TO 20:NEXT DE:RETURN
400 FOR N=YP0+2 TO 104:POKE 704,N:SOUND 0,N+144,14,4:SOU
ND 1,N+145,10,4:FOR Q=1 TO 10:NEXT Q
410 D=USR(ADR(JUMP$),PM0+N,PM1):NEXT N:SOUND 1,0,0,0:YP0
=N-1
420 NO=INT(21*RND(1)+60):FOR T=15 TO 0 STEP -1:SOUND 0,N
O,2,T:FOR DE=1 TO 5:NEXT DE:NEXT T:SOUND 0,0,0,0:GOTO 330
500 SOUND 0,72,10,4:GOSUB 391:SOUND 0,64,10,4:GOSUB 391:
SOUND 1,72+1,10,2:GOSUB 391
501 SOUND 0,60,10,4:GOSUB 391:SOUND 1,64+1,10,2:GOSUB 39
1:SOUND 0,50,10,4:GOSUB 392
502 SOUND 1,60+1,10,2:GOSUB 391:SOUND 0,60,10,4:GOSUB 39
1:SOUND 1,50+1,10,4:GOSUB 391
503 SOUND 0,42,10,4:GOSUB 393
515 SOUND 0,0,0,0:SOUND 1,0,0,0
520 ? #6;CHR$(125):D=USR(ADR(JUMP$),PM0+YP0,PMB+100):FOR
N=1 TO 3:D=USR(ADR(JUMP$),PM(N)+YP(N),PMB+100):NEXT N
530 ROUND=ROUND+1:VEL=VEL+(VEL<3):GOSUB 1000:SOUND 1,0,0
,0:SOUND 2,0,0,0
540 SCORE=SCORE+1000:NUMBLOCK=0
550 GOTO 340
600 REM
630 ? CHR$(125):POKE 656,0:POKE 657,0:? "GAME OVER":POKE
656,0:POKE 657,20:? "SCORE ";SCORE
640 POKE 656,1:POKE 657,0:? "press FIRE to play"
650 IF STRIG(0)=1 THEN 650
660 ? #6;CHR$(125):ROUND=1:VEL=1:SCORE=0:NUMBLOCK=0:LIVE
S=3:SHAPE=PM1
670 GOSUB 1000:SOUND 1,0,0,0:SOUND 2,0,0,0:GOTO 340
800 GRAPHICS 4+16:POKE 710,0:POKE 708,28:COLOR 1
810 CHAR$="GIL-BERT"
820 I0=PEEK(560)+PEEK(561)*256:I1=PEEK(I0+4)+PEEK(I0+5)*256
830 FOR U=1 TO LEN(CHAR$)
840 I2=57344+((ASC(CHAR$(U,U))-32)*8):I3=I1+PY*40+PX+U-1
850 FOR Z=0 TO 7:POKE I3+Z*10,PEEK(I2+Z):NEXT Z:NEXT U
860 FOR T=0 TO 150:NEXT T
865 FOR N=1 TO 14:READ A,B,C:SOUND 0,A,10,6:PLOT A,B:DRA
WTO A,C:PLOT A+1,B:DRAWTO A+1,C:NEXT N:SOUND 0,0,0,0
870 DATA 33,24,28,35,22,30,37,20,23,37,28,34,39,20,23,39
,26,30,39,34,34
```

GIL-BERT

```
875 DATA 41,20,30,43,20,23,43,28,34,45,22,25,45,28,30,45
,34,34,47,24,28
876 POKE 752,1:? "   Bounce on BLOCKS to change their
    colour and AVOID the SKULLS..."
877 ? "          Game by M+B CHRISTIAN"
880 PMB=PEEK(106)-24:POKE 54279,PMB:PMB=PMB*256:POKE 623
,1
890 POKE 704,34:POKE 705,6:POKE 706,172:POKE 707,250:DIM
 JUMP$(26),PM(4),XP(4),YP(4)
900 FOR J=1 TO 25:READ AA:JUMP$(J,J)=CHR$(AA):NEXT J
910 DATA 104,104,133,204,104,133,203,104,133,207,104,133
,206,160,0,177,206,145,203,200,192,50,208,247,96
918 FOR Z=1 TO 4
919 FOR N=555 TO 0 STEP -5:SOUND 0,N,10,4:SOUND 1,0+X,10
,4
920 S=S+1:SETCOLOR 0,S,8:NEXT N:NEXT Z:POKE 88,0
921 POKE 89,PEEK(106)-24:? #6;CHR$(125)
925 SOUND 0,0,0,0:SOUND 1,0,0,0
930 FOR LOOP=2 TO 8 STEP 3:FOR J=0 TO 7:READ AA:POKE PMB
+LOOP*10+J,AA:NEXT J:NEXT LOOP
940 DATA 60,126,211,155,255,126,36,108
945 DATA 60,126,203,217,255,126,36,54
950 DATA 28,42,73,109,42,28,34,28
960 FOR N=1 TO 3:PM(N)=PMB+512+128*N:YP(N)=-8+82*(N=1 OR
 N=3):NEXT N:XP(1)=55:XP(2)=135:XP(3)=160
970 PM0=PMB+512:XP0=115:YP0=-2:PM1=PMB:PM2=PMB+30:PM3=PM
B+60
990 RETURN
1000 SETCOLOR 0,RND(0)*15,8:SETCOLOR 1,RND(0)*15,4:COLOR
 1:X=89:Y=-10:N=-2:ZW=0
1010 FOR M=1 TO 7
1020 N=N+1:X=X-6:Y=Y+10
1030 FOR Q=-1 TO N
1040 XX=X+Q*12
1050 PLOT XX,Y:DRAWTO XX+6,Y+4:DRAWTO XX,Y+8:DRAWTO XX-6
,Y+4:DRAWTO XX,Y
1060 PLOT XX-6,Y+4:DRAWTO XX-6,Y+10:DRAWTO XX,Y+14:DRAWT
O XX,Y+8:PLOT XX,Y+14:DRAWTO XX+6,Y+10:DRAWTO XX+6,Y+4
1070 ZW=ZW+4:SOUND 1,ZW*1.6+60+1,10,4:SOUND 2,ZW*1.6+60,
10,4:COLOR 2
1080 PLOT XX-5,Y+5:DRAWTO XX-5,Y+10:PLOT XX-4,Y+6:DRAWTO
 XX-4,Y+10:PLOT XX-3,Y+7:DRAWTO XX-3,Y+11
1082 PLOT XX-2,Y+8:DRAWTO XX-2,Y+12:PLOT XX-1,Y+8:DRAWTO
 XX-1,Y+14:COLOR 1:PLOT XX+1,Y+8:DRAWTO XX+1,Y+14
1084 PLOT XX+2,Y+8:DRAWTO XX+2,Y+12:PLOT XX+3,Y+7:DRAWTO
 XX+3,Y+11:PLOT XX+4,Y+6:DRAWTO XX+4,Y+10
1086 PLOT XX+5,Y+5:DRAWTO XX+5,Y+10:COLOR 1:NEXT Q:NEXT
M
1088 RETURN
1090 SOUND 1,0,0,0:SOUND 2,0,0,0:DL=PEEK(560)+256*PEEK(5
61):NUMBLOCK=0:SHAPE=PM1:BOARD=1:VEL=1
1100 A=DL+85
1110 POKE A,71:POKE A+3,6:POKE A+4,6:POKE A+5,65:POKE A+
6,PEEK(A+7):POKE A+7,PEEK(A+8):POKE 710,12
1120 POKE 656,0:POKE 657,22:? "score    ROUND"
1130 POKE 656,0:POKE 657,2:? SCORE;:POKE 657,15:? ROUND;
1140 POKE 656,1:POKE 657,7:? "LIVES:";LIVES;:D=USR(ADR(J
UMP$),PM0+YP0,PM1):POKE 53248,XP0
1150 FOR Q=1 TO 3:D=USR(ADR(JUMP$),PM(Q)+YP(Q),PM3):POKE
 53248+Q,XP(Q):NEXT Q:RETURN
```

LABEL MAKER

Continued from 17.

```
5671 ? CHR$(125);? ;? DAT$;? "CONT":POSITION 0,0
5710 POKE 842,13:STOP
5720 POKE 842,12
5750 POKE 559,SON:? CHR$(125):RETURN
6620 REM DATA GENERATOR
6635 OFF=0:SON=PEEK(559):POKE 559,OFF
6640 FOR L=1 TO LEN(DAT$) STEP LEN(DAT$)/3
6670 ? CHR$(125);?
6680 ? DAT$(L,L+(LEN(DAT$)/3-1))
6690 ? "CONT"
6700 POSITION 0,0
6710 POKE 842,13:STOP
6720 POKE 842,12:NEXT L:GOSUB 5647:POKE 559,SON
6751 POKE 82,2:POKE 83,39:RETURN
6810 REM PRINT OUT ROUTINE
6830 TRAP 6861:FOR L=1 TO 210 STEP LEN(TEXT$)/7
6840 LPRINT "    ";TEXT$(L,L+(LEN(TEXT$)/7-1));
6850 LPRINT "    ";TEXT$(L,L+(LEN(TEXT$)/7-1))
6860 NEXT L:FOR LOOP=1 TO 3:LPRINT :RETURN
6861 ? CHR$(253);:POSITION 11,18:? " PRINTER NOT THERE ";
FOR DEL=1 TO 500:NEXT DEL
6862 POSITION 11,18:? "                    ":RETURN
6961 ? CHR$(253);:POSITION 10,18:? " REFERENCE DUPLICATED
 ":FOR DEL=1 TO 500:NEXT DEL
6962 ? CHR$(253);:POSITION 10,18:? "
 "
6963 POP :GOTO 570
6964 ? CHR$(253);:POSITION 10,18:? " ABORTED OUT OF MEMOR
Y":FOR DEL=1 TO 500:NEXT DEL
6965 ? CHR$(253);:POSITION 10,18:? "
 "
6966 POP :GOTO 570
7010 REM RETRIEVE  ROUTINE
7030 RESTORE 11000:TEXT$="":CHECK$=""
7031 POSITION 21,4:? "REF >>";:INPUT CHECK$
7032 POSITION 21,4:? "RETRIEVE";CHECK$
7040 READ TEMP1$,TEMP2$,TEMP3$
7050 IF TEMP1$(2,3)="##" THEN GOTO 7210
7060 IF TEMP3$(69,71)=CHECK$ THEN 7210
7070 GOTO 7040
7210 TEXT$(LEN(TEXT$)+1)=TEMP1$(2,71)
7220 TEXT$(LEN(TEXT$)+1)=TEMP2$(2,71)
7230 TEXT$(LEN(TEXT$)+1)=TEMP3$(2,71)
7240 RETURN
7999 REM LISTING ROUTINE
8000 RESTORE 11000
8005 TEXT$=""
8010 READ TEMP1$,TEMP2$,TEMP3$:GOSUB 7210:GOSUB 3861:GOS
UB 690:GET #1,CHAR:GOSUB 700
8011 IF CHAR<>27 THEN IF TEMP1$(2,3)<>"##" THEN 8005
8020 RETURN
10000 DATA 11000,1
32701 DATA "###,.++ LABEL PRINTER ++,.####---- --------+++
++---------- SORRY THE"
32702 DATA " LABEL YOU REQUESTED    IS NOT IN THE MEMORY
    ---------------------"
32703 DATA "----------REMEMBER TO PRESS THE 'S' KEY TO S
TORE YOUR LABEL IN MEMORY "
```

DE-TOKENISER

by Ron Levy

If you read the article 'BASICALLY SECURE' elsewhere in this issue you will realise how some of those BASIC programs that you have,but are unable to look at are 'protected', or hidden. If you would like to examine these programs and see how they work (of course you would not intend to copy them!!) then here is just the program for you.

DE-TOKENISER will take your protected program, chew it up, and spit out a readable version. It is ideally used in a disk system , but can be used in a cassette-only system provided that you realise the constraints in not being able to have simultaneous input and output cassette files. The other problem with cassette programs is that in order for DE-TOKENISER to input them using BASIC's GET #1,X command then they need to have long Inter-Record Gaps. Inter-Record Gaps are the pauses in between the 128 byte bursts of data in a cassette load or save operation. The reason for having two different sizes of Inter-Record Gaps (let's call it IRG) is that whereas data files are saved and loaded at infrequent intervals, programs are either 'CLOAD'ed or 'CSAVE'd in one continuous stream.

Take, for example, a BASIC program which PUTs or PRINTs data to a cassette file. When the 128 byte file buffer is filled the program is paused while the operating system dumps the 128 byte buffer to the cassette recorder. It may have been a while since the last time the buffer was dumped, so the cassette motor will probably be stationary and will need time to accelerate to the correct speed. Hence the long IRG's. Since CLOAD and CSAVE operations are performed by fast machine code routines there is no need for a pause, and consequently short IRG's can be and are used to save time and tape length.

A Token Gesture

In order to understand what DE-TOKENISER does, you will need to understand how a BASIC program is held in memory.

There are two main methods of storing a BASIC program in a computer's memory. The first method is that which most users will be familiar with and this is as ASCII text. Here, the BASIC program is stored in the computer exactly as it is typed in by the programmer, and would usually be stored to cassette or disk in the same form. This method is not ideal for the computer since it has to analyse the text and perform its syntax error checking during running. The second method is the one used by ATARI BASIC, and is a far more efficient one; it is called TOKENISING.

The principle is to take the line of the BASIC program as it is typed in, and convert each element into a more compact and logical form. For example, a line number can have a value anywhere in the range of 0 to 32767 and when stored in ASCII form (as the user would read it) it will occupy between 1 and 5 bytes of memory. Not by coincidence, 32767 (the largest line number that BASIC will allow) is also the largest number that can be represented by a 16 bit binary word. Thus two bytes of memory can be used to efficiently store any line number.

This principle of 'pre-digestion' of the line number is applied to the entire program line, and I will now explain more fully how the ATARI BASIC tokenises and stores its programs.

Following the line number there must be a command. Instead of storing this, for example, as REM, PRINT, or INPUT etc., each command has a single byte number assigned to it and this is stored in its place. For example, REM has a value of 0, PRINT has a value of 32, and INPUT has a value of 2. The third major item is called the OPERATOR. The term OPERATOR actually covers a wide range of items, and includes mathematical and string equates, numeric and string variables, and several algebraic operations. The last major group is the FUNCTION tokens. These are the ones which, when executed, will return a value to the program. Study Table 1 for Commands, Operators and Functions and their corresponding token values.

The first thing you should notice from this table is that whereas COMMANDS and OPERATORS are considered by

COMMANDS	OPERATORS	FUNCTIONS
00 0 REM	0E 14 [NUMERIC CONSTANT]	3D 61 STR$
01 1 DATA	0F 15 [STRING CONSTANT]	3E 62 CHR$
02 2 INPUT	10 16 [NOT USED]	3F 63 USR
03 3 COLOR	11 17 [NOT USED]	40 64 ASC
04 4 LIST	12 18 ,	41 65 VAL
05 5 ENTER	13 19 $	42 66 LEN
06 6 LET	14 20 : [END OF STATEMENT]	43 67 ADR
07 7 IF	15 21 ;	44 68 ATN
08 8 FOR	16 22 [END OF LINE]	45 69 COS
09 9 NEXT	17 23 GOTO	46 70 PEEK
0A 10 GOTO	18 24 GOSUB	47 71 SIN
0B 11 GO TO	19 25 TO	48 72 RND
0C 12 GOSUB	1A 26 STEP	49 73 FRE
0D 13 TRAP	1B 27 THEN	4A 74 EXP
0E 14 BYE	1C 28 #	4B 75 LOG
0F 15 CONT	1D 29 <= [NUMERICS]	4C 76 CLOG
10 16 COM	1E 30 <> "	4D 77 SQR
11 17 CLOSE	1F 31 >= "	4E 78 SGN
12 18 CLR	20 32 < "	4F 79 ABS
13 19 DEG	21 33 > "	50 80 INT
14 20 DIM	22 34 = "	51 81 PADDLE
15 21 END	23 35	52 82 STICK
16 22 NEW	24 36 ^ "	53 83 PTRIG
17 23 OPEN	25 37 + "	54 84 STRIG
18 24 LOAD	26 38 - "	
19 25 SAVE	27 39 / "	
1A 26 STATUS	28 40 NOT	
1B 27 NOTE	29 41 OR	
1C 28 POINT	2A 42 AND	
1D 29 XIO	2B 43 (	
1E 30 ON	2C 44)	
1F 31 POKE	2D 45 = [ARITHMETIC ASSIGN]	
20 32 PRINT	2E 46 = [STRING ASSIGN]	
21 33 RAD	2F 47 <= [STRINGS]	
22 34 READ	30 48 <> "	
23 35 RESTORE	31 49 >= "	
24 36 RETURN	32 50 < "	
25 37 RUN	33 51 > "	
26 38 STOP	34 52 = "	
27 39 POP	35 53 + [UNARY]	
28 40 ?	36 54 - "	
29 41 GET	37 55 ([STRING LEFT PARENTHESIS]	
2A 42 PUT	38 56 ([ARRAY LEFT PARENTHESIS]	
2B 43 GRAPHICS	39 57 ([DIM ARRAY LEFT PARENTHESIS]	
2C 44 PLOT	3A 58 ([FUNCTION LEFT PARENTHESIS]	
2D 45 POSITION	3B 59 ([DIM STR LEFT PARENTHESIS]	
2E 46 DOS	3C 60 , [ARRAY COMMA]	
2F 47 DRAWTO		
30 48 SETCOLOR		
31 49 LOCATE		
32 50 SOUND		
33 51 LPRINT		
34 52 CSAVE		
35 53 CLOAD		
36 54 [IMPLIED 'LET']		
37 55 ERROR - [FOR SYNTAX]		

Table 1

BASIC as being completely different types of tokens, FUNCTIONS are merely an additional range of OPERATORS (and are treated as such by BASIC). Let's look in detail now at how BASIC actually stores a program line. Consider the following line:

```
10 LET A =1:PRINT X
```

This is how it will look when it has been tokenised:

```
 0 0A       >
 1 00       >[LINE NUMBER]
 2 13        [LINE OFFSET]
 3 0F        [STATEMENT OFFSET]
 4 06 LET    [COMMAND]
 5 80 A      [VARIABLE]
 6 2D =      [OPERATOR]
 7 0E        [OPERATOR- NUMERIC CONSTANT FOLLOWS]
 8 40       >
 9 01       >
10 00       >[CONSANT 1]
11 00       >
12 00       >
13 00       >
14 14 :     [OPERATOR- END OF STATEMENT]
15 13       [STATEMENT OFFSET]
16 20 PRINT [COMMAND]
17 B1 X     [VARIABLE]
18 16       [OPERATOR- END OF LINE]
```

There are a few aspects of this token line which may be a little unclear or obscure. The LINE OFFSET is the offset of the first byte of the next program line from the first byte of the current line. The STATEMENT OFFSET byte is the offset of the first byte of the next statement from the first byte of the current line.

Variables

Another question which should also arise is how does BASIC differentiate between COMMANDS/OPERATORS/FUNCTIONS and variables in a program line. What happens is that as each new variable is entered by the programmer it is assigned its own one byte number, beginning with the number 128 (80 HEX). The greatest COMMAND/OPERATOR/FUNCTION number used is 84 (54 HEX), and consequently there is no confusion between these and variables. It is also worth noting that this variable reference-number scheme is in fact the reason that you can only have up to 128 variable names in a program.

Numbers

The OPERATOR 14 (ØE HEX) is used to indicate that directly following is a number. These are stored in the ATARI computer as a 6 byte floating point number only, and the 6 bytes are used as follows. Byte one is the exponent, and its value is also used to indicate whether the mantissa and the exponent are positive or negative. The next 5 bytes are used to store the 10 digit mantissa, and it does this in BCD format.

Strings

Strings are the only parts of a BASIC program that are stored exactly as they are entered by the programmer, and BASIC signals that one is to follow by using the OPERATOR number 15 (ØF HEX). Strings can, however, be of varying length and could contain any character with a byte value in the range of Ø to 255. Because of this BASIC needs to know precisely how long the string is. This is accomplished by storing a single byte before the string whose value indicates the length of the string. The following example shows how this will look in practice:

```
10 A$="TEXT"
```

The tokenised version is:

```
 0 0A    >[LINE NUMBER
 1 00    >
 2 0E     [LINE OFFSET]
 3 0E     [STATEMENT OFFSET]
 4 36     [COMMAND- IMPLIED 'LET']
 5 80 A$  [VARIABLE]
 6 2E =   [OPERATOR]
 7 0F     [OPERATOR- STRING CONSTANT FOLLOWS]
 8 04     [STRING LENGTH INDICATOR]
 9 54 T  >
10 45 E  >[STRING DATA]
11 58 X  >
12 54 T  >
13 16     [OPERATOR- END OF LINE]
```

The Invisible LET

The first part of any statement has to be a COMMAND, followed by a set of OPERATORS, FUNCTIONS, variables and constants. If this is the case, then how can we write a program line that begins with a variable? For example, 10 A$=B$ is a line which not only begins with a variable in place of a command, but also does not even appear to contain a COMMAND! The answer lies in the IMPLIED 'LET', COMMAND byte 54 (36 HEX). What happens is that as the line is entered by the programmer, BASIC assumes it to mean 10 LET A$=B$, but instead of using COMMAND byte 6 (explicit LET), it uses byte 54.The tokenised line is then identical to one with an explicit LET as far as the BASIC treats it during the program run, but when BASIC LISTs the line it will not print the word 'LET' to the screen.

The Program

The first task of the program is to create its COMMAND and OPERATOR strings and arrays. Note that I have included the FUNCTIONS into the OPERATOR string. As ATARI BASIC does not provide string arrays I have had to use long strings and create pointer arrays for the positions of the individual sub-strings. It is possible to alter the COMMAND data table, as I will explain later, but for now these should be typed precisely as shown.

When RUNning, the program will, at times, need to convert numbers from their internal 6 byte floating point representation into printable ASCII form, and this is done in a very crafty manner. As the 6 bytes are retrieved they are POKEd into the locations used to store the value of the variable 'N' in the program that is running. 'N' is then used and its value is printed. Lines 60 to 76 are the routine which finds the location in memory of 'N' and it keeps this in the variable 'NLOC' for use later.

Lines 100 to 190 establish the input and output files. Input can be from cassette or disk, and ouput can be cassette, disk , printer, or in fact any legitimate output device. If you intend using a disk system, or you are using cassette system and dumping the output to the screen only (by typing RETURN in response to output file) or printer (P:), then there is no limitation on the source program size as you are not attempting to use two cassette files concurrently. If, however, you wish to input from cassette, then there is a problem! Since the output dump must be performed when the entire loading and de-tokenising has finished, the produced program listing must be stored in memory. This is accomplished by creating CAS$, which has been dimensioned to as large as possible (allowing for some immediate mode operations should you stop the program). When de-tokenising has finished the input file is closed, the cassette is OPENed for output, and CAS$ is printed to it.

Lines 200 to 270 load in and store the 14 bytes of file header. This contains the pointers which BASIC uses to find the various tables, and DE-TOKENISER uses these to extract the variable names and the variable value tables.

Lines 400 to 490 will now load the variable names and store them in VAR$. This will be printed to the screen, and you will be asked whether you wish to use this table. You will have to

decide for yourself whether you think this is corrupted or not, although protected programs will often have this filled with carriage returns (CHR 155), and this will then be quite obvious. If you type N then the program will build a new variable name table for you. It does this while it is reading in the variable value table in lines 500 to 800.

Next is the heart of the program, lines 1000 to 1990. This is where the tokens are examined and the source coding created, and it will require careful examination in order to follow precisely how it operates. Basically (no pun intended) the main re-entry points as de-tokenising progresses are line 1000, where a new line is expected, line 1140 where a new statement is started from (i.e. after a line number or a colon), and 1230 where the next OPERATOR is to be read in. The program continues in this section until 'end of file' is reached for the input file, at which point a trap to line 20000 occurs.

From line 2000 on, there is a collection of subroutines which are called regularly by the main program. These are well labelled and so do not require explaining in detail.

```
 0 REM ==============================
 1 REM =.........DETOKENISER.........=
 2 REM =..A BASIC Programme Cracker..=
 3 REM =.........By RON LEVY.........=
 8 REM ==============================
10 DIM F1$(15),F2$(15),CMD$(300),OPR$(200),A$(20),N$(10)
,VAR$(400),OPT$(9),D$(256)
11 DIM CMD(100,1),OPR(100,1),VAR(127,1)
20 PRINT CHR$(125);"   DE-TOKENISER.........By Ron Levy"
22 PRINT "   ==============================":PRINT
30 REM ..Create CMD and OPR strings..
31 REM ..=========================..
32 FOR I=0 TO 55
33 READ A$:A=LEN(A$)
34 L=LEN(CMD$)+1:CMD(I,0)=L:CMD(I,1)=L+A-1
36 CMD$(L,L+A)=A$
38 NEXT I
39 PRINT "Commands:":PRINT CMD$:PRINT
40 FOR I=18 TO 84
41 IF I=18 OR I=60 THEN A$=",":GOTO 43
42 READ A$
43 A=LEN(A$)
44 L=LEN(OPR$)+1:OPR(I,0)=L:OPR(I,1)=L+A-1
46 OPR$(L,L+A)=A$
48 NEXT I
49 PRINT "Operators:":? OPR$:PRINT
60 REM ..Find variable 'N' data loc..
61 REM ..=========================..
62 VNTP=PEEK(130)+PEEK(131)*256
63 VNTD=PEEK(132)+PEEK(133)*256
64 VVTP=PEEK(134)+PEEK(135)*256
65 FOR I=VNTP TO VNTD
66 A=PEEK(I):C=C+1
68 IF A=206 AND C=1 THEN 76
70 IF A>127 THEN V=V+1:C=0
72 NEXT I
74 PRINT "SYSTEM ERROR.....":? " Variable N Cannot Be lo
cated !!!!":STOP
76 NLOC=VVTP+V*8+1:CAS=1
100 REM .......OPEN The Files........
101 REM .......==============........
110 PRINT "INPUT file....";:INPUT F1$
120 PRINT F1$
130 PRINT "OUTPUT file...";:INPUT F2$
140 PRINT F2$
150 IF F2$="" OR F2$="C:" THEN 170
160 OPEN #2,8,0,F2$:GOTO 190
170 IF F2$="" THEN 190
180 CSZ=FRE(0)-128:DIM CAS$(CSZ)
190 OPEN #1,4,0,F1$
200 REM .......Get File Header.......
201 REM .......==============.......
210 GET #1,X:GET #1,Y:LOMEM=X+256*Y
220 GET #1,X:GET #1,Y:VNTP=X+256*Y
230 GET #1,X:GET #1,Y:VNTD=X+256*Y
240 GET #1,X:GET #1,Y:VVTP=X+256*Y
250 GET #1,X:GET #1,Y:STMTAB=X+256*Y
260 GET #1,X:GET #1,Y:STMCUR=X+256*Y
270 GET #1,X:GET #1,Y:STARP=X+256*Y
280 ? "LOMEM  = ";LOMEM
282 ? "VNTP   = ";VNTP,"VNTD =";VNTD
284 ? "STMTAB = ";STMTAB
400 REM ...Get Variable Name Table...
401 REM ...======================...
402 PRINT "Existing Variable Name Table......."
405 C=0:USV=0:POKE 766,1
410 FOR I=VNTP TO VNTD
420 GET #1,X:C=C+1
430 PRINT CHR$(X);
440 VAR$(C,C)=CHR$(X)
450 NEXT I:PRINT :PRINT
470 PRINT "Use Existing VARIABLE NAMES <Y/N>..";
480 INPUT OPT$
485 IF OPT$<>"Y" THEN VAR$=""
490 IF OPT$="Y" THEN USV=1:GOSUB 16000
500 REM ......Get Variable Data......
501 REM ......================......
505 C=0:VN=0:VN1=0:VA=0:VA1=0:VS=0:VS1=0
510 FOR I=VVTP TO STMTAB-1
515 C=C+1:IF C>8 THEN C=1
520 GET #1,X
600 IF USV=1 THEN 800
610 IF C<>1 THEN 800
620 IF X<>0 THEN 680
630 VN=VN+1:IF VN>26 THEN VN1=VN1+1:VN=1
640 N$=CHR$(VN+64):IF VN1<>0 THEN N$(2,2)=CHR$(VN1+176)
650 IF VN1=0 THEN N$=CHR$(VN+192)
670 GOTO 790
680 IF X>66 THEN 740
690 VA=VA+1:IF VA>26 THEN VA1=VA1+1:VA=1
700 N$=CHR$(VA+64):IF VA1<>0 THEN N$(2,2)=CHR$(VA1+48):N
$(3,3)=CHR$(168)
710 IF VA1=0 THEN N$(2,2)=CHR$(168)
720 GOTO 790
740 VS=VS+1:IF VS>26 THEN VS1=VS1+1:VS=1
750 N$=CHR$(VS+64):IF VS1<>0 THEN N$(2,2)=CHR$(VN1+48):N
$(3,3)=CHR$(164)
760 IF VS1=0 THEN N$(2,2)=CHR$(164)
790 L=LEN(VAR$)+1:VAR$(L,L+LEN(N$))=N$
800 NEXT I
810 IF USV=1 THEN 890
820 PRINT "New Variable Name Table:"
830 PRINT VAR$:PRINT
890 IF USV=0 THEN GOSUB 16000
900 TRAP 20000
1000 REM .......Get Token Line.......
1001 REM .......==============.......
1100 LC=3
1110 GET #1,X:GET #1,Y:L=X+256*Y
1120 D$=STR$(L):GOSUB 10000:D$=" ":GOSUB 10000
```

```
1130 GET #1,LO:REM Line Offset.
1140 GET #1,SO:REM Statement Offset.
1150 LC=LC+1:REM Inc. Line Counter.
1200 GET #1,X:LC=LC+1:REM Command.
1205 IF X=0 OR X=1 THEN D$=CMD$(CMD(X,0),CMD(X,1)):GOSUB
 10000:GOSUB 4000:GOSUB 10000:GOTO 1000
1206 IF X=54 THEN D$="":GOTO 1220
1210 D$=CMD$(CMD(X,0),CMD(X,1))
1220 GOSUB 10000
1230 GET #1,X:LC=LC+1
1240 IF X=22 THEN D$=CHR$(155):GOSUB 10000:GOTO 1000
1250 IF X=20 THEN D$=":":GOSUB 10000:GOTO 1140:REM Next
Statement.
1260 IF X<>27 THEN 1270
1261 D$=" THEN ":GOSUB 10000
1262 IF SO=LC THEN 1140
1263 GOTO 1230
1270 IF X=14 THEN GOSUB 2000:D$=STR$(N):GOSUB 10000:GOTO
 1230
1280 IF X=15 THEN GOSUB 3000:GOSUB 10000:GOTO 1230
1290 IF X>127 THEN X=X-128:D$=VAR$(VAR(X,0),VAR(X,1)):GO
SUB 10000:GOTO 1230
1295 IF X=56 OR X=57 THEN 1230
1300 D$=OPR$(OPR(X,0),OPR(X,1))
1310 GOSUB 10000
1990 GOTO 1230
2000 REM ....Get Number Constant.....
2001 REM ....==================.....
2010 FOR N1=1 TO 6
2015 GET #1,X:LC=LC+1
2020 POKE N1+NLOC,X
2030 NEXT N1
2090 RETURN :REM Done ================
3000 REM ....Get String Constant.....
3001 REM ....==================.....
3005 D$=CHR$(34)
3010 GET #1,X:LC=LC+1
3015 IF X=0 THEN N1=2:GOTO 3050
3020 FOR N1=2 TO X+1
3030 GET #1,X:D$(N1,N1)=CHR$(X)
3040 LC=LC+1:NEXT N1
3050 D$(N1,N1)=CHR$(34)
3090 RETURN :REM Done ================
4000 REM ......Get REM Command......
4001 REM ......==============......
4010 N1=1:D$=""
4020 GET #1,X:D$(N1,N1)=CHR$(X)
4025 LC=LC+1
4030 N1=N1+1:IF X<>155 THEN 4020
4090 RETURN :REM Done ================
10000 REM ......Output Routine.......
10001 REM ......==============.......
10050 IF D$="" THEN 10900
10100 PRINT D$;
10200 IF F2$="" OR F2$="C:" THEN 10400
10300 PRINT #2;D$;
10400 IF F2$<>"C:" THEN 10900
10410 X=LEN(D$):IF X+CAS>CSZ THEN PRINT :PRINT "Memory S
ize Exceeded.....";CHR$(253):STOP
10420 CAS$(CAS,CAS+X-1)=D$
10440 CAS=LEN(CAS$)+1
10900 RETURN :REM Done ===============
16000 REM Create V-Name pointer array
16010 REM ============================
16020 N1=1:C=0
16100 FOR I=1 TO LEN(VAR$)
16140 A=ASC(VAR$(I,I))
16150 IF A<128 THEN 16190
16160 VAR(C,0)=N1:VAR(C,1)=I
16170 N1=I+1:C=C+1
16180 VAR$(I,I)=CHR$(A-128)
16190 NEXT I
16290 RETURN :REM Done.===============
20000 REM .....End Of File Trap......
20001 REM .....================......
20100 PRINT :IF F2$<>"C:" THEN 20190
20110 PRINT "Insert OUTPUT tape into RECORDER,"
20120 PRINT "Press PLAY and RECORD,"
20130 PRINT "Then hit <RETURN>.....";
20150 OPEN #2,8,0,F2$
20160 PRINT #2;CAS$
20190 CLOSE #2
20500 PRINT CHR$(253)
20510 PRINT ".............Complete................."
20900 STOP
32000 DATA REM ,DATA ,INPUT ,COLOR ,LIST ,ENTER ,LET ,IF
 ,FOR ,NEXT ,GOTO ,GO TO ,GOSUB ,TRAP ,BYE,CONT
32001 DATA COM ,CLOSE ,CLR,DEG,DIM ,END,NEW,OPEN ,LOAD ,
SAVE ,STATUS ,NOTE ,POINT ,XIO ,ON ,POKE ,PRINT ,RAD
32002 DATA READ ,RESTORE ,RETURN,RUN ,STOP,POP,? ,GET ,P
UT ,GRAPHICS ,PLOT ,POSITION ,DOS
32003 DATA DRAWTO ,SETCOLOR ,LOCATE ,SOUND ,LPRINT ,CSAV
E,CLOAD,LET ,ERROR -
32010 DATA $,:,;,,GOTO ,GOSUB , TO , STEP ,THEN ,#,<=,<>
,>=,<,>,=, ,^,+,-,/, NOT , OR , AND ,(,),=,=,<=,<>,>=,<
32011 DATA >,=,+,-,(,(,(,(
32012 DATA STR$,CHR$,USR,ASC,VAL,LEN,ADR,ATN,COS,PEEK,SI
N,RND,FRE,EXP,LOG,CLOG,SQR,SGN,ABS,INT,PADDLE,STICK
32013 DATA STICK,STRIG
```

Using the Program

If you are using a cassette system and are using the facility to store the output program in CAS$, then the size of program that can be dealt with may be limited by the amount of free memory that you have. If this becomes a problem, however, it is possible to drastically reduce the size of the output program by using abbreviations for the commands. Just as you can type L. or GR. in your own program to save typing the full commands, LIST and GRAPHICS; the same thing can be done to the output program by altering the data tables that are read in to create the command table. Type the following lines and keep them 'LIST'ed to cassette or disk file, and 'ENTER' them over the main program when you have memory size problems.

32000DATA REM,D.,I.,C.,L.,E.,LET,IF,F.,N.,G.,GO TO,
GOS.,T.,B.,CONT
32001 DATA COM,CL.,CLR,DEG,DIM,END,NEW,O.,LO.,
S.,ST.,NO.,P.,X.,ON,POKE,PR.,RAD
32002 DATA READ,RES.,RET.,RUN,STOP,POP,?,GET,PUT,
GR.,PL.,POS.,DOS
32003 DATA DR.,SE.,LOC.,SO.,LP.,CS.,CLOAD,LET,ERROR -

You could also gain a little more memory space by removing the remarks in the de-tokeniser program, as these are only there to assist you in understanding and modifying it to your own needs.

Something which you should notice when running the program is that there is always a line 32768, and it is normally a SAVE command. The reason for this is that BASIC treats immediate mode commands by first giving them this line number before executing them, and of course the last command given by the programmer would be one to SAVE his program!

If you only have a cassette system, then to create a CSAVE'd test program with long IRG's you must use SAVE"C:".

Happy cracking!!

QRA

by Chris Barlow

This program is not a game. So what is QRA? The letters QRA come from the International 'Q' Code, similar to the C.B. 'Ten' code, 10-10, 10-4, etc, The Q code gives quick information when in radio communication with other stations. This code is not in common use by the C.B. fraternity, but more commonly used by Amateur Radio enthusiasts worldwide. The code 'QRA' means 'Please give me your exact location', similar to the C.B. '10-20'. It is easy for Radio Amateurs on V.H.F. and U.H.F. bands to communicate to most of Europe, distances up to 2500 kms. are not uncommon under favourable atmospheric conditions.

QRA CODE

Well, what is all of this to you, a computer enthusiast? It appears that several of you, like myself, are licensed Radio Amateurs, my 'call' is G8LVK. Radio Amateurs use the request QRA to ascertain the distance between their station and the station being contacted. Many enthusiasts take part in contests in a given period of time. Points are calculated on the distance of each station contacted. The QRA Code for a station consists of two letters, for the first Large Square, (see Fig. 1). Each Large Square is sub-divided into eighty smaller squares, numbered from 01 to 80 (top left to bottom right - see Fig. 2). To obtain a precise fix, each smaller square is further sub-divided into nine smaller squares. These smaller squares are lettered A-J (as shown in Fig. 3). Therefore, a complete QRA position might be AL34G, which is my own QRA location for Leigh-on-Sea, Essex.

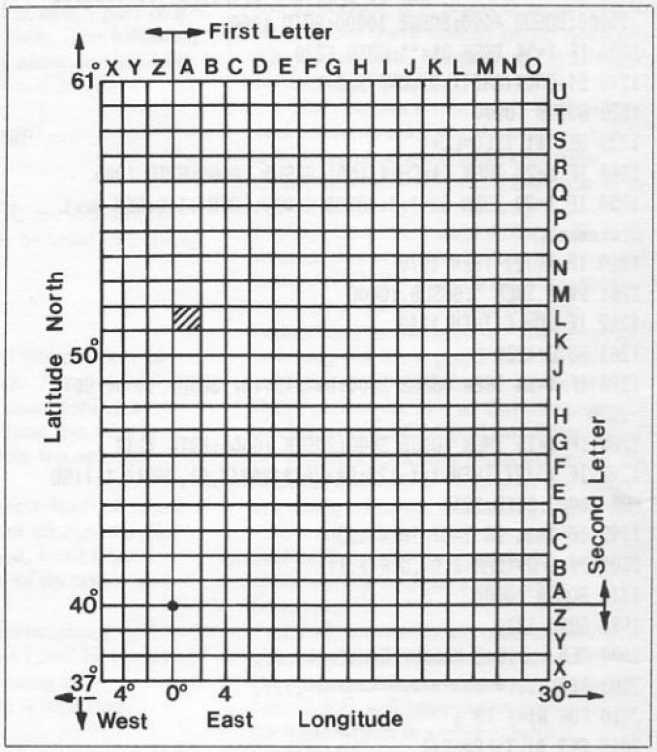


Figure 1. Large Square

QRA BREAKDOWN

To calculate the QRA position, one needs to convert Latitude and Longitude into Large and Small Squares. This is not easy, so why do it? The QRA position data is not as accurate as Latitude and Longitude. The reason for using the less accurate Code Square is that when in radio contact with other stations, under difficult reception conditions, there is a possibility of misunderstanding the more complex structure of Latitude and Longitude reference, i.e. 52 degrees 13 minutes 26 seconds North, 01 degrees 03 minutes 16 seconds East (Where??). The idea of using a computer to calculate distances between stations is not new. Several programs have been published in the past, but none for the Atari (it is only a games machine after all!!). This version was devised from a program written by D.W. Hughes which appeared in the June 1982 edition of Radio Communications, the monthly journal of the Radio Society of Great Britain. That program was written in standard Microsoft Basic. To convert the program into Atari Basic was quite straightforward. The main alterations involved changing the string handling, also extra colour and sound routines were added.

My version of the program is quite easy to use. The first piece of information you must enter is your own QRA reference (the BASE STATION), and then enter the QRA reference of the station you have been given. If you input an invalid QRA code an error

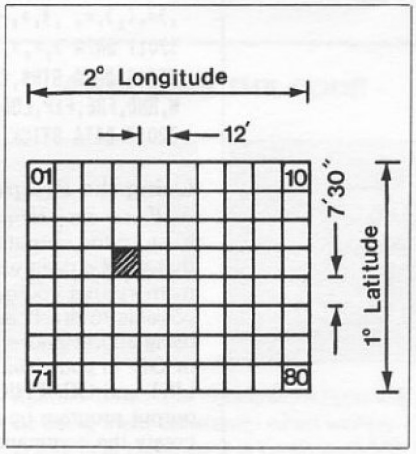


Figure 2. Small Square

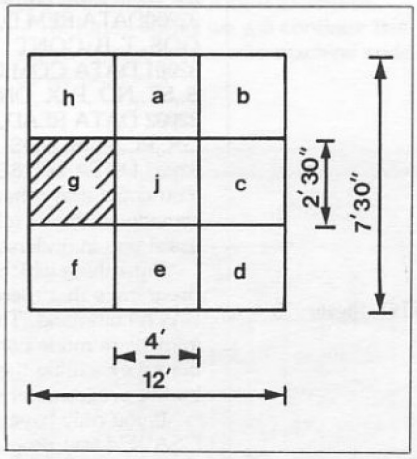


Figure 3. Smaller Square

'INVALID QRA' will be displayed. If it becomes necessary to change the Base Station reference whilst running the program, simply input the word 'BASE'. This will allow a new Base Reference to be entered. This last function is useful if you are operating portable or mobile or when another station requests information from you if he does not have such a program, or computer, himself.

Note, program lines 50, 110, 220 and 590 all have 37 spaces.

To assist you in verifying the program results, see Fig. 4 for some examples.

Base	Contact	Distance (kms)	Point Value
AL34G	PN45C	2045	81
AL34G	AJ27B	210	9
ZM01J	PR02E	2081	83
ZL80A	YZ47G	1314	53
XR40D	HB05D	2193	87

Figure 4. Verification Examples

Also shown is a map of Europe overlaid with a grid marked off with the start of the QRA codes.

This program has been extremely useful to me and I hope that you will also gain some pleasure in using it. Hopefully, some of you good people with interests in Amateur Radio will forward similar programs of interest to the Software Library.

QRA

```
10 CLR :GRAPHICS 0:POKE 752,1:SETCOLOR 2,0,0:SETCOLOR 4,
12,7
20 POSITION 4,1:PRINT "xxxxxxxxxx G3LVK, xxxxxxxxxx"
30 GOSUB 610
40 DIM Z$(5):R=6371.02:PI=3.14159265
50 POSITION 33,5:PRINT "
   "
60 POSITION 4,5:PRINT " ENTER QRA OF BASE STATION. ";:IN
PUT Z$
70 IF LEN(Z$)=0 THEN 60
80 GOSUB 270
90 IF K=1 THEN 50
100 X0=X:Y0=Y:Z0=Z
110 POSITION 33,7:PRINT "
   "
120 SOUND 0,0,0,0:POSITION 4,7:PRINT " ENTER QRA OF CONT
ACT..... ";:INPUT Z$
130 IF LEN(Z$)=0 THEN 120
140 IF Z$="BASE" THEN 50
150 GOSUB 270
160 IF K=1 THEN 110
170 SOUND 0,45,10,10
180 DX=X-X0:DY=Y-Y0:DZ=Z-Z0
190 Q=SQR(DX*DX+DY*DY+DZ*DZ):Q=Q/(2*R)
200 KM=2*R*ATN(Q/SQR(1-Q*Q))
210 KM=INT(KM+0.5):SC=INT(KM/50):SC=SC*2+1
220 POSITION 4,9:PRINT "
   "
230 POSITION 4,9:PRINT Z$;" : ";KM;" KM.";" : ";SC;
240 IF SC<2 THEN PRINT " POINT."
250 IF SC>1 THEN PRINT " POINTS."
260 POSITION 33,7:PRINT "       ":GOTO 120
270 K=0:IF LEN(Z$)<5 THEN 540
280 A=ASC(Z$(1,1))-65
290 B=ASC(Z$(2,2))-65
300 I=ASC(Z$(3,3))-48
310 J=ASC(Z$(4,4))-48
320 C=ASC(Z$(5,5))-65
330 IF A<0 OR A>26 THEN 540
340 IF B<0 OR B>26 THEN 540
350 IF C<0 OR C>9 OR C=8 THEN 540
360 IF I<0 OR I>8 THEN 540
370 IF J<0 OR J>9 THEN 540
380 IF I=8 AND J>0 THEN 540
390 IF A>17 THEN A=A-26
400 LO=A*2:IF B>21 THEN B=B-26
410 LA=B+40:II=I:JJ=J:IF J<>0 THEN 430
420 II=II-1:JJ=10
430 LO=LO-0.1+(0.2*JJ):LA=LA+0.9375-(0.125*II)
440 IF C=7 OR C=6 OR C=5 THEN DO=0.03333333
450 IF C=0 OR C=9 OR C=4 THEN DO=0.1
460 IF C=1 OR C=2 OR C=3 THEN DO=0.16666666
470 LO=LO+DO
480 IF C=5 OR C=4 OR C=3 THEN DA=0.02083
490 IF C=6 OR C=9 OR C=2 THEN DA=0.0625
500 IF C=7 OR C=0 OR C=1 THEN DA=0.10417
510 LA=LA+DA:LO=LO*PI/180:LA=LA*PI/180
520 SN=SIN(LO):SA=SIN(LA):CO=COS(LO):CA=COS(LA):X=R*CO*C
A:Y=R*SN*CA:Z=R*SA
530 RETURN
540 K=1:SETCOLOR 2,3,7:SETCOLOR 4,3,7
550 POSITION 2,11:PRINT "xxxxxx ERROR xxxxxx INVALID QRA
 CODE."
560 POKE 53768,1:POKE 53760,60:POKE 53761,175:FOR T=0 TO
 50:NEXT T
570 POKE 53760,255:FOR T=0 TO 80:NEXT T:SOUND 0,0,0,0
580 FOR T=0 TO 100:NEXT T
590 POSITION 2,11:PRINT "
   "
600 SETCOLOR 2,0,0:SETCOLOR 4,12,7:RETURN
610 FOR LOOP=0 TO 17:READ D:POKE 1536+LOOP,D:NEXT LOOP
620 MAC=USR(1536):RETURN
630 DATA 104,169,6,162,6,160,11,32,92
640 DATA 228,96,169,0,133,77,76,95,228
```

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

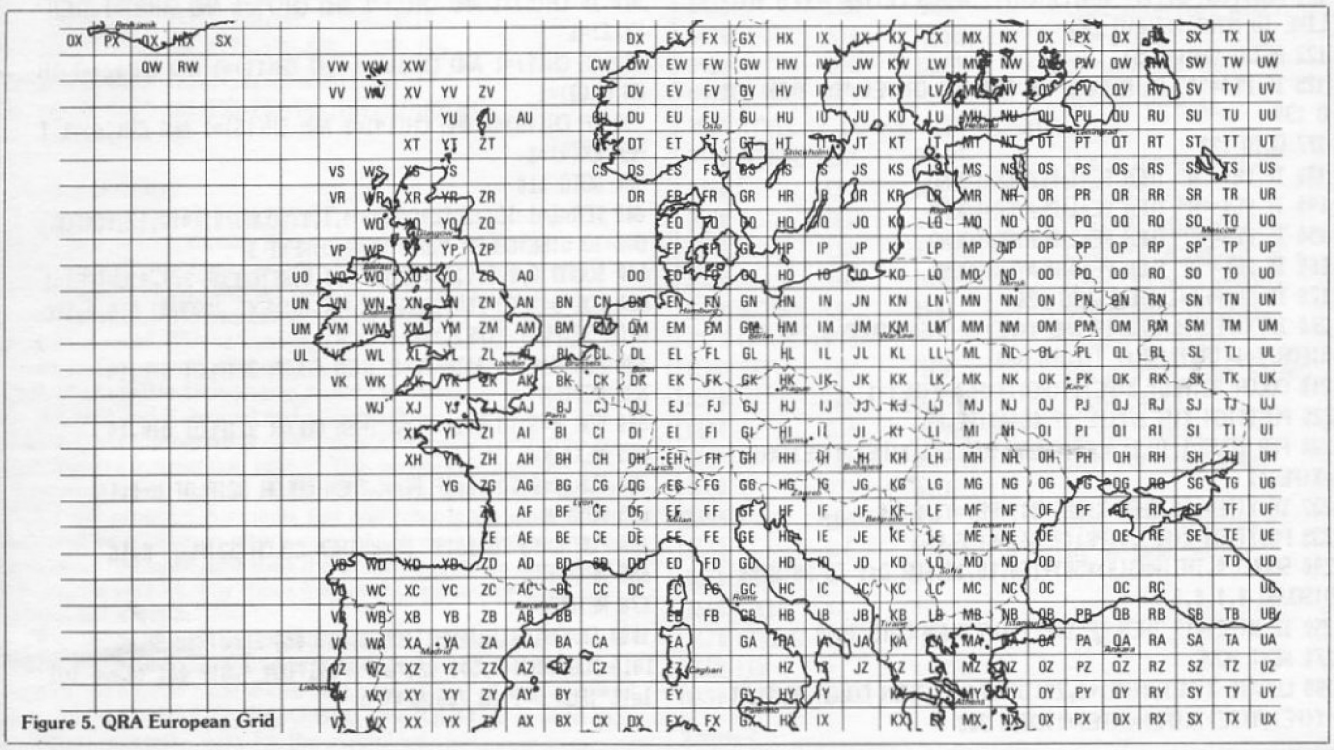


Figure 5. QRA European Grid

ELECTRIC SHOCK

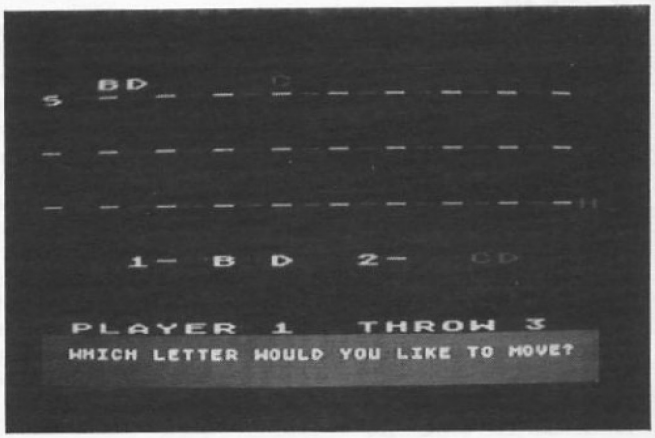
Runs in 16K Cassette or Disk

By Steve Tullett - Dalkeith

This is a game for 2 to 4 players, the object is to get your four men home before your opponents'. The only problem is that there are hidden 'electrified' plates in your path and if your man steps on one, well you can guess what happens! Use the keyboard to move your men the number of spaces randomly selected by the computer. This is an entertaining game for all the family.

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

```
10 GOSUB 1000
20 DIM A$(17),X(17),Y(17),EX(17),CH(17),PX(9),PY(9),PE$(1)
30 FOR P=1 TO 9 STEP 4:FOR I=0 TO 2:R=INT(RND(0)*20)+1:P
X(CT)=R:PY(CT)=P:CT=CT+1:NEXT I:NEXT P
40 A$="ABCDABCDabcdabcd"
50 FOR PE=1 TO 16:X(PE)=0:Y(PE)=1:CH(PE)=0:EX(PE)=0:NEXT PE
55 IF N=2 THEN EX(3)=EX(4)=1
60 IF N=3 THEN EX(4)=1
70 GRAPHICS 1:FOR L=1 TO 3:? #6:? #6
80 PRINT #6;"--------------------"
90 NEXT L:POSITION 0,2:? #6;"S":POSITION 19,10:? #6;"H"
100 POSITION 3,14:? #6;"1-ABCD  2-ABCD":POSITION 3,16:?
#6;"3-abcd  4-abcd"
102 IF N=2 THEN POSITION 3,16:? #6;"                "
103 IF N=3 THEN POSITION 10,16:? #6;"        "
105 POSITION 1,19:? #6;"PLAYER ":POSITION 11,19:? #6;"TH
ROW ":? :POKE 752,1
110 FOR PLAYER=1 TO N:SOUND 0,0,0,0
111 IF EX(1)=1 AND EX(2)=1 AND N=2 THEN GOSUB 3000
112 IF EX(1)=1 AND EX(2)=1 AND EX(3)=1 AND N=3 THEN GOSU
B 3000
115 IF EX(1)=1 AND EX(2)=1 AND EX(3)=1 AND EX(4)=1 AND N
=4 THEN GOSUB 3000
117 IF EX(PLAYER)=1 THEN GOTO 360
118 PLACES=INT(RND(0)*6)+1:POSITION 17,19:? #6;PLACES
120 POSITION 8,19:? #6;PLAYER:? "WHICH LETTER WOULD YOU
LIKE TO MOVE";:INPUT PE$
122 PRINT CHR$(125)
125 IF PE$="A" OR PE$="B" OR PE$="C" OR PE$="D" THEN GOT
O 130
127 GOTO 120
130 IF PE$="A" THEN PE=0+PLAYER+ROUND
140 IF PE$="B" THEN PE=1+PLAYER+ROUND
150 IF PE$="C" THEN PE=2+PLAYER+ROUND
160 IF PE$="D" THEN PE=3+PLAYER+ROUND
170 IF CH(PE)=1 THEN GOTO 120
200 IF X(PE)+PLACES>19 AND Y(PE)=9 THEN SOUND 0,240,12,1
5:FOR I=1 TO 90:NEXT I:GOTO 360
210 COLOR 32:PLOT X(PE),Y(PE):Y(PE)=Y(PE)-1
220 POSITION X(PE),Y(PE):? #6;A$(PE,PE)
230 FOR MOVE=1 TO PLACES:COLOR 32:PLOT X(PE),Y(PE):X(PE)
=X(PE)+1
232 IF X(PE)=20 THEN X(PE)=0:Y(PE)=Y(PE)+4
235 POSITION X(PE),Y(PE):? #6;A$(PE,PE)
240 SOUND 0,INT(RND(0)*50)+150,10,15:FOR D=1 TO 30:NEXT
D:SOUND 0,0,0,0
250 IF X(PE)=19 THEN IF Y(PE)+1=9 THEN GOTO 2000
270 NEXT MOVE
280 LOCATE X(PE),Y(PE)+1,ZZ:IF ZZ<>32 THEN COLOR 32:PLOT
 X(PE),Y(PE):X(PE)=X(PE)+1:GOTO 300
```




```
290 GOTO 320
300 IF X(PE)=19 THEN IF Y(PE)+1=9 THEN GOTO 2000
310 IF X(PE)=20 THEN X(PE)=0:Y(PE)=Y(PE)+4
311 POSITION X(PE),Y(PE):? #6;A$(PE,PE)
312 SOUND 0,INT(RND(0)*50)+150,10,15:FOR D=1 TO 30:NEXT
D:SOUND 0,0,0,0:GOTO 280
320 COLOR 32:PLOT X(PE),Y(PE):Y(PE)=Y(PE)+1
330 POSITION X(PE),Y(PE):? #6;A$(PE,PE)
340 FOR CT=0 TO 8:IF X(PE)=PX(CT) AND Y(PE)=PY(CT) THEN
GOSUB 600:GOTO 360
350 NEXT CT
360 ROUND=ROUND+3:IF PLAYER=N AND N=2 THEN ROUND=0:GOTO
390
370 IF PLAYER=N AND N=3 THEN ROUND=0:GOTO 390
380 IF PLAYER=4 AND N=4 THEN ROUND=0
390 NEXT PLAYER
400 IF CH(1)=1 AND CH(2)=1 AND CH(3)=1 AND CH(4)=1 THEN
EX(1)=1
410 IF CH(5)=1 AND CH(6)=1 AND CH(7)=1 AND CH(8)=1 THEN
EX(2)=1
420 IF CH(9)=1 AND CH(10)=1 AND CH(11)=1 AND CH(12)=1 TH
EN EX(3)=1
430 IF CH(13)=1 AND CH(14)=1 AND CH(15)=1 AND CH(16)=1 T
HEN EX(4)=1
450 GOTO 110
600 FOR I=1 TO 15:SETCOLOR 4,I,I:SOUND 0,I*10,12,15:FOR
D=1 TO 5:SETCOLOR 4,D,D:NEXT D:NEXT I
610 SOUND 0,0,0,0:SETCOLOR 4,0,0:SETCOLOR 2,9,4:CH(PE)=1
620 FOR J=1 TO 200:SOUND 0,J,8,15:NEXT J:SOUND 0,0,0,0:C
OLOR 32:PLOT X(PE),Y(PE)
630 FOR K=1 TO 4:IF PE=K THEN COLOR 32:PLOT 4+K,14
635 NEXT K
640 FOR K=5 TO 8:IF PE=K THEN COLOR 32:PLOT 8+K,14
645 NEXT K
650 FOR K=9 TO 12:IF PE=K THEN COLOR 32:PLOT K-4,16
655 NEXT K
660 FOR K=13 TO 16:IF PE=K THEN COLOR 32:PLOT K,16
665 NEXT K
670 RETURN
1000 GRAPHICS 18:POSITION 3,2:? #6;"ELecTRic SHock!"
1010 POSITION 9,5:? #6;"BY":POSITION 4,8:? #6;"steve tul
lett":FOR D=1 TO 700:NEXT D
```

Continued on 28.

DRAGONFIRE

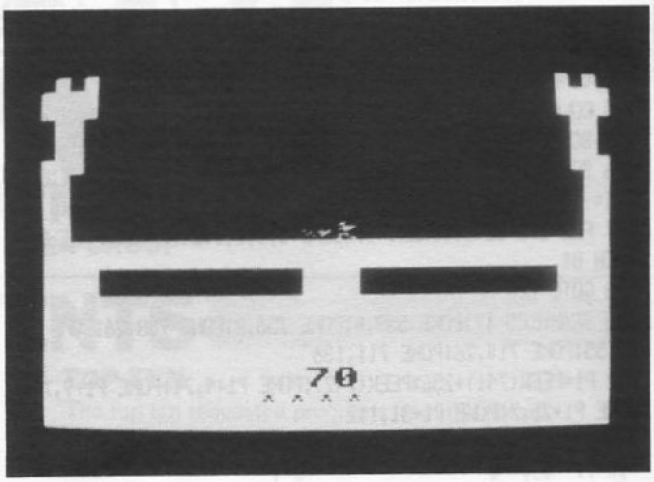
Runs in 16K Cassette or 32K Disk

by Grahame Fairall

Can you dodge the Dragon's fiery breath and get across the drawbridge to the treasure rooms? Collect as much treasure as you can whilst you avoid the attack of the ferocious Dragon. Use your joystick to make your man leap or duck the flaming fireballs.

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

```
 10 DIM A$(2),M$(32),W$(9),P(2),F(6),FY(6)
 12 GRAPHICS 0:POKE 559,0
 15 A$="[\"
 18 FOR I=1 TO 7:READ DAT:W$(I,I)=CHR$(I):NEXT I
 19 DATA 137,170,11,140,13,174,143
 20 P=PEEK(106)-8:OLD=224*256
 30 FOR I=1 TO 32:READ A:M$(I,I)=CHR$(A):NEXT I
 35 DATA 104,104,133,204,104,133,203,104,133,206,104,133,
205,162,4,160,0,177,203,145,205,136,208,249,230,204
 40 DATA 230,206,202,208,240,96
 45 X=USR(ADR(M$),OLD,P*256)
 50 FOR I=8 TO 127:READ A:POKE P*256+I,A:NEXT I
 55 FOR I=128 TO 207:POKE P*256+I,255-(PEEK(OLD+I)):NEXT I
 60 FOR I=208 TO 263:READ A:POKE P*256+I,A:NEXT I
 65 FOR I=472 TO 511:READ A:POKE P*256+I,A:NEXT I
 66 POKE 708,54:POKE 709,15:POKE 710,0:POKE 711,24:POKE 7
52,1:POKE 82,2:IF SC>HI THEN HI=SC
 67 L=PEEK(741)+256*PEEK(742)+6:FOR I=4 TO 8:POKE L+I,4:N
EXT I:POKE L+10,6:POKE L+14,7
 68 POSITION 2,0:? "SCORE         HIGH SCORE":POSITION 3,1
:? SC:POSITION 18,1:? HI
 69 ? :? :? "## ##  ### ### ### # # ### # ## ###";:? "
# # # # # # #  # # ## # #  # # # # ";
 70 ? "# # ##  ### # # # # #### ## # ## ## ";:? "# # #
# # # # # # # # ## #  # # # # ";
 71 ? "## # # # # ### ### # # #  # # # ###":POSITION 1
,10:? "BY GRAHAME FAIRALL":POKE 82,0
 72 POSITION 26,11:? "PRESS SELECT FOR DIFFICULTY":POSITI
ON 10,15:? "PRESS START TO PLAY":DIF=1:POKE 559,34
 73 POSITION 29,13:? DIF
 74 IF PEEK(53279)=5 THEN FOR I=50 TO -50 STEP -10:SOUND
0,ABS(I),10,10:NEXT I:DIF=DIF+1:IF DIF>6 THEN DIF=1
 75 IF PEEK(53279)=6 THEN 77
 76 SOUND 0,0,0,0:GOTO 73
 77 SK=DIF-0.1:SC=0:LI=5:DD=10-DIF:SL=DIF*2:IF DD>5 THEN
DD=5
 80 GOSUB 200:X=18:DO=0:MO=1:F(1)=1:F(2)=13:F(3)=7:F(4)=1
0:F(5)=16:F(6)=4:FOR I=1 TO 6:FY(I)=RND(0)+8:NEXT I
 90 S=STICK(0):X1=(S=9)+(S=10)+(S=11):IF S=13 AND DO=0 TH
EN DO=DD
 91 POSITION 9,19:? #6;SC:IF STRIG(0)=0 AND UP=0 THEN UP=
DD
 92 IF X1<>0 THEN DO=0:UP=0:IF X-X1<=0 THEN 130
 93 IF X1<>0 THEN COLOR 0:PLOT X,8:PLOT X,9:X=X-X1:COLOR
6:PLOT X,8:COLOR 7:PLOT X,9:SC=SC+5:GOTO 100
 94 IF DO=0 AND UP=0 THEN 100
 95 IF DO<>0 THEN COLOR 0:PLOT X,8:COLOR 8:PLOT X,9:DO=DO
-1
 96 IF UP<>0 THEN COLOR 0:PLOT X,9:COLOR 30:PLOT X,8:UP=U
P-1
 97 IF DO=0 AND UP=0 THEN COLOR 6:PLOT X,8:COLOR 7:PLOT X
,9
```




```
100 FOR I=1 TO SK:LOCATE F(I),FY(I),Z:COLOR 0:PLOT F(I),
FY(I):IF Z<>0 AND Z<>32 AND Z<>28 AND Z<>29 THEN 120
101 F(I)=F(I)+MO:LOCATE F(I),FY(I),Z:IF Z<>0 AND Z<>32 T
HEN 120
102 COLOR 28+RND(0):PLOT F(I),FY(I):IF F(I)>=17 THEN COL
OR 0:PLOT F(I),FY(I):F(I)=1:FY(I)=RND(0)*1.4+8
103 NEXT I
110 SOUND 0,F(1),8,10:SOUND 1,F(2),8,10:GOTO 90
120 SOUND 1,0,0,0:COLOR 28:PLOT F(I),FY(I)
121 FOR I=1 TO 6:FOR J=50 TO 0 STEP -5:SOUND 0,J,10,I+4:
NEXT J:NEXT I:FOR I=10 TO 40:SOUND 0,I,8,10
122 SOUND 1,I+2,8,10:NEXT I:SOUND 0,40,8,2:SOUND 1,42,8,
2:COLOR 0:PLOT X,8
124 Z=0:FOR I=10 TO 14:COLOR Z:PLOT X,I-1:LOCATE X,I,Z:C
OLOR 96:PLOT X,I:FOR J=1 TO 30:NEXT J:NEXT I
126 SOUND 1,0,0,0:J=11:FOR I=20 TO 10 STEP -0.3:J=J-0.3:
SOUND 0,I,8,J:FOR K=0 TO 10:NEXT K:NEXT I
127 SOUND 0,0,0,0:LI=LI-1:FOR I=0 TO 200:NEXT I:IF LI>0
THEN 80
128 POSITION 5,5:? #6;"game  over":FOR I=0 TO 255 STEP 2
:POKE 709,I:SOUND 0,I,10,10:SOUND 1,255-I,10,10
129 SOUND 2,I/2+64,8,10:SOUND 3,255-(I/2+64),8,10:NEXT I
:FOR I=0 TO 3:SOUND I,0,0,0:NEXT I:POKE 559,0:GRAPHICS 0
:GOTO 66
130 FOR I=-30 TO 30 STEP 2:SOUND 0,ABS(I)+50,10,10:SOUND
 1,ABS(I)+40,10,10:NEXT I
140 GOSUB 300
150 S=STICK(0):X1=(S=5)+(S=6)+(S=7)-(S=9)-(S=10)-(S=11):
Y1=(S=5)+(S=9)+(S=13)-(S=6)-(S=10)-(S=14)
151 IF X+X1<1 THEN 195
152 COLOR 0:PLOT X,Y:X=X+X1:Y=Y+Y1:LOCATE X,Y,W:IF W<>91
 AND W<>92 AND W<>186 AND W<>0 AND W<>32 THEN 169
154 IF W<>0 AND W<>32 AND W<>91 AND W<>92 OR Y>18 THEN X
=X-X1:Y=Y-Y1
156 COLOR 31+(X1<0)*-23:PLOT X,Y:IF CO=SL THEN GOSUB 190
160 Q1=(Q+1<X)-(Q+1>X):POSITION Q,19:? #6;"  ":Q=Q+Q1:P
OSITION Q,19:? #6;A$
162 FOR I=1 TO SK:COLOR Z:PLOT F(I),FY(I):FY(I)=FY(I)-MO
:LOCATE F(I),FY(I),Z:IF FY(I)<=3 THEN FY(I)=18:F(I)=Q+1
163 IF FY(I)<Y THEN FY(I)=18:F(I)=Q+1
164 COLOR 126+RND(0):PLOT F(I),FY(I):IF Z=93 OR Z=94 THE
N Z=0
```

DRAGONFIRE

```
165 IF X=F(I) AND Y=FY(I) THEN 196
166 NEXT I:SOUND 2,0,0,0:IF RND(0)<0.01 THEN COLOR 140:P
LOT RND(0)*16+2,RND(0)*11+4
168 SOUND 0,F(1),8,10:SOUND 1,F(2),8,10:GOTO 150
169 IF W=93 OR W=94 THEN 196
170 CO=CO+1:SOUND 2,20,10,14:GOTO 156
190 COLOR 0:PLOT 0,8:DRAWTO 0,13:SOUND 2,0,0,0:RETURN
195 SC=SC+CO*10:GOTO 80
196 SOUND 2,0,0,0:SOUND 1,0,0,0:COLOR 96:PLOT X,Y:J=0:FO
R I=-12 TO 12 STEP 0.2:J=J+1:SOUND 0,J,8,ABS(I)+3:NEXT I
197 FOR I=0 TO 300:NEXT I:SOUND 0,0,0,0:LI=LI-1:IF LI>0
THEN 80
198 GOTO 128
200 GRAPHICS 17:POKE 559,0:POKE 756,P:POKE 708,26:POKE 7
09,55:POKE 710,26:POKE 711,136
202 P1=PEEK(741)+256*PEEK(742):POKE P1+4,71:POKE P1+9,7:
POKE P1+25,7:POKE P1+31,112
210 ? #6;"!!                ZZ";:? #6;"$$
 $$";:? #6;""$                "$";
212 ? #6;""$                "$";:? #6;"$$
 "$"
214 COLOR 164:PLOT 0,5:DRAWTO 0,10:DRAWTO 19,10:DRAWTO 1
9,5:PLOT 0,17:DRAWTO 0,11:DRAWTO 19,11:DRAWTO 19,17
216 DRAWTO 0,17:PLOT 1,12:DRAWTO 1,16:PLOT 18,12:DRAWTO
18,16
218 FOR I=18 TO 22:PLOT 0,I:DRAWTO 19,I:NEXT I
220 COLOR 132:FOR I=14 TO 16:PLOT 2,I:DRAWTO 17,I:NEXT I
:COLOR 164:PLOT 9,12:DRAWTO 9,16:PLOT 10,12:DRAWTO 10,16
222 COLOR 187:FOR I=1 TO LI:PLOT I+6,20:NEXT I:POSITION
9,19:? #6;SC
224 COLOR 8:PLOT 18,9
230 POKE 559,34:RETURN
300 GRAPHICS 17:POKE 559,0:POKE 756,P:POKE 708,214:POKE
709,52:POKE 710,8:POKE 711,24
302 P1=PEEK(741)+256*PEEK(742):POKE P1+27,7:POKE P1+33,1
12
303 SL=SL+1:SK=SK+0.1
304 COLOR 186:PLOT 0,2:DRAWTO 19,2
308 PLOT 0,0:DRAWTO 0,19:PLOT 19,0:DRAWTO 19,19:COLOR 16
4:FOR I=0 TO 1:PLOT 0,I:DRAWTO 19,I:NEXT I
310 FOR I=20 TO 23:PLOT 0,I:DRAWTO 19,I:NEXT I
320 COLOR 187:FOR I=1 TO LI:PLOT I+6,22:NEXT I:POSITION
9,21:? #6;SC
330 FOR J=1 TO SL:I=I+1:X=RND(0)*16+2:Y=RND(0)*11+4:LOCA
TE X,Y,Z:IF Z<>0 AND Z<>32 THEN J=J-1:I=I-1:NEXT J
332 COLOR ASC(W$(I,I)):PLOT X,Y:IF I=7 THEN I=0
334 NEXT J:Q=12:CO=0
336 FOR I=1 TO SK:F(I)=RND(1)*16+2:FY(I)=RND(1)*11+4:LOC
ATE F(I),FY(I),ZZ:IF ZZ<>0 AND ZZ<>32 THEN I=I-1:NEXT I
338 NEXT I
342 FOR J=1 TO SK:X=RND(0)*16+2:Y=RND(0)*11+4:LOCATE X,Y
,Z:IF Z<>0 AND Z<>32 THEN J=J-1:NEXT J
344 NEXT J
346 X=INT(RND(0)*10+5):Y=INT(RND(0)*10+5):LOCATE X,Y,Z:I
F Z<>0 AND Z<>32 THEN CO=CO+1
350 POKE 559,34:RETURN
9000 DATA 15,15,15,15,255,255,255,255,15,15,15,15,15,15,
15,15,240,240,240,240,240,240,240,240
9001 DATA 255,255,255,255,255,255,255,255,240,240,240,24
0,255,255,255,255,0,0,0,0,0,0,48,48
9002 DATA 30,60,24,0,56,40,38,98,96,126,28,56,24,56,44,1
00,40,170,170,170,254,16,16,56
9003 DATA 0,120,188,222,239,112,55,23,153,189,189,255,66
,66,102,102,54,28,50,111,71,111,99,62
9004 DATA 0,60,24,60,110,223,110,60,135,157,189,255,60,1
6,16,56,0,36,0,129,0,90,90,126
9005 DATA 255,255,255,255,199,199,199,199,255,255,253,24
8,253,250,250,255,0,0,42,143,13,166,0,0
9006 DATA 0,0,133,47,70,20,0,0,24,31,14,156,255,1,0,0,6,
126,56,28,24,28,52,38
9007 DATA 0,0,0,139,255,60,204,4,112,128,123,255,255,95,
77,225,224,48,234,255,176,142,0,128
9008 DATA 0,0,0,0,0,0,0,0,212,122,191,102,17,68,40,130,2
4,52,110,60,145,68,17,72
```

Continued from 26.

ELECTRIC SHOCK

```
1020 FOR I=1 TO 15:FOR C=8 TO 15:SETCOLOR 4,15-I,0:SETCO
LOR 2,C,8:SOUND 0,C*12,12,C
1030 NEXT C:NEXT I:FOR J=1 TO 200:SOUND 0,J,8,15:NEXT J:
SOUND 0,0,0,0
1040 GRAPHICS 0:SETCOLOR 4,3,0:SETCOLOR 2,12,4:POKE 752,
1:? :? "This is a simple game for 2-4 players.";
1050 ? "Each player has 4 letters (ABCD) of"
1060 ? "the same colour. The computer throws  a die for
each turn and the player"
1065 ? "choses which of his letters to move."
1070 ? "Letters progress from the START (S)   on the top
 line to HOME (H) at the"
1080 ? "bottom. But beware ......."
1090 ? :? "3 squares on each level are 'LIVE'.   If you
land on one at the end of a"
1100 ? "move the shock you receive destroys   your playi
ng letter."
1110 ? :? "The first player to land exactly on   HOME wi
ns. If a landing place is"
1120 ? "occupied your letter moves forward to the next a
vailable space."
1130 ? :? :? "HOW MANY PLAYERS FOR THE GAME ";
1140 INPUT N
1145 IF N<2 OR N>4 THEN GOTO 1130
1150 RETURN
2000 COLOR 32:PLOT X(PE),Y(PE):POSITION X(PE),Y(PE)+1:?
#6;A$(PE,PE)
2005 FOR I=255 TO 0 STEP -1:SOUND 0,I,10,15:FOR D=0 TO 2
:NEXT D:NEXT I:SOUND 0,0,0,0
2010 GRAPHICS 18:POSITION 3,3:? #6;"player ";PLAYER;" wi
ns"
2020 POSITION 3,6:? #6;"to play again
        press START"
2030 IF PEEK(53279)=6 THEN RUN
2040 SETCOLOR 2,INT(RND(1)*16),10:GOTO 2030
3000 GRAPHICS 18:POSITION 5,3:? #6;"ALL LOSE!"
3010 GOTO 2020
```

THE U.K.

ATARI

COMPUTER OWNERS CLUB

INDEPENDENT USER GROUP

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