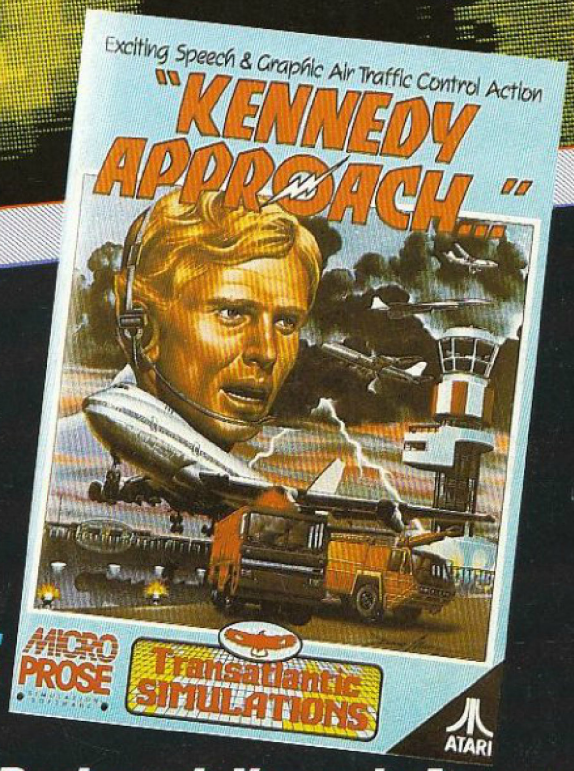
The U.K. ATARI Computer Owners Club Issue 10 Price £1.00

Independent User Group

Monitor



Digitised Pictures
Find out the facts inside!

MIDI Language explained!

Action packed issue including:
Disk Jacket program
Comms for Beginners
PCB Paranoia, 3D Maze
2 in 1 Competition

Reviewed: Kennedy Approach
Great American Road Race
Wishbringer, Red Moon

ST Rules O.K.

What a tremendous impact the 520ST has had on the computing world. It must be the most written about computer over the last six months, with nearly all being highly favourable! And now the 520ST has taken the 'PC of the Year' award, a truly astounding feat considering that there is hardly any software available as yet. Even the software that was to be supplied with the machine is only just arriving. The official version of ST Basic has been released as well as GemDraw and GemWrite. Atari are also freely passing around ST Atariwriter in the 'Public Domain'. Software is building slowly though and more and more will be coming along I'm sure. In the States it is claimed that between 30,000 and 55,000 ST systems have been sold. Seems an awful lot to me, but lets hope its true!

On the subject of software, US Gold have brought out the computer version of 'The Goonies' which is currently doing good business at the cinema. It is available on disk (£14.95) and cassette (£9.95). Also Maplin Electronics are selling off all their Atari software at silly prices (I mean £3.95 and £4.95), mind you most of them are fairly old titles, but there are some good bargains nontheless. Their catalogue, which is sold in W.H. Smith's for £1.45, has all the titles listed, but I am told that stocks are limited so get in quick!

Last issue we introduced 'Monitor Disks' and these have been a great success. Obviously many of you don't like typing in much. As those of you who have had disks will know, they contain extra goodies not mentioned before, as well as the listings in the magazine, many of the assembler listings which we had no room to print have been included on the disks. On Monitor 10 disk, as there are not quite as many listings as usual, we have included 'Chase', which is at the top of our software chart at the moment, and is an excellent game, as an extra bonus.

Finally, we would like to wish you all a very merry Christmas and we hope that in the new year you will continue to enjoy Monitor Magazine as much as you seem to now. Thanks for all your letters of praise, its nice to know that we are getting the formula somewhere near right!

CONTENTS

CREDITS

Editor	Roy Smith
Art Editor	Peter Blackmore
Technical Editor	Ron Levy
Technical Editor	Keith Mayhew
Adventure Editor	Steve Hillen

SUBSCRIPTION RATES

U.K. and Ireland	£4.00
Europe	£7.00
Outside Europe (Surface)	£7.00
Outside Europe (Airmail)	£10.50

The above prices are in English Pounds Sterling and include postage and packing.

A subscription/membership fee to join the U.K. Atari Computer Owners Club is just £4.00 for four issues of the club magazine. All cheques/postal orders are to be made payable to the 'U.K. Atari Computer Owners Club'. Overseas membership is also available at slightly higher rates. Overseas members who use the Library service should include enough extra monies to cover return postage.

ADVERTISEMENTS

Please note that the club cannot be held legally responsible for claims made by advertisers.

CLUB ADDRESS: The U.K. Atari Computer Owners Club, P.O. Box 3, Rayleigh, Essex.

CRACKING THE CODE

by Keith Mayhew Part Six

Last time, we looked at the assembler and saw how it alleviated us from many monotonous tasks, the most simple of which are looking up opcodes for mnemonics, and values for labels – such as the location of a variable or the address of an instruction in the program. Also we do not have to bother ourselves with the fact that two-byte addresses are stored in reverse order and that branches use a relative offset: the assembler handles them all for us and does not make mistakes.

However, this does not mean that you should forget about the subtleties of how instructions are stored in memory and their effect on memory locations and registers when executed. Such an understanding will be found to be invaluable when developing programs, mainly because it will give you confidence in what you are doing, but also, it will give you an in-depth view of what your program does when running; this will help most when you are trying to debug your code, especially at the machine code level of opcodes and operand bytes.

We have already studied each machine instruction in isolation in earlier articles, so you should now be in a position to write any program you like and you can even use an assembler to help you. Well, that is not the whole story: although we have seen each instruction and how to write it in assembly language form, we have not yet finished with ways of addressing data in memory; without these vital addressing modes, which involve one of the index registers, virtually no real program could be written! Do not worry though, we shall look at these offending modes later and once they are understood, all you will lack is some experience. Of course, experience will only come with practice, for which there is no substitute, but you will find that this will help you choose the appropriate set of instructions to perform a certain task. It is the ability to pick the optimum set of instructions that will make your program more efficient, that is it will run faster, be smaller or use less memory space for its data; it is very difficult to find the best of these three, and in certain cases one may be more important than another. You need not concern yourself to deeply about optimising your program, the fact that it works will usually be of more importance to you!

A Case of Efficiency

Although your first attempt at a program may work, you may also find that it is large, slow and that trying to work out how the program actually does what it does could be something of a nightmare! Often clarity in a program is the most important item, this will be achieved by careful planning, whether it is on paper or in your head. A clearly coded program will need less comments to steer you through it when you are trying to change or debug it; remember that if you do not have a good approach to writing programs you will probably end up in such a tangled mess that debugging it will be nearly impossible. If your programs head this way then it is usually worth having a re-write before it becomes out of hand. Writing clear programs will only come with practice and you will develop your own style of writing, just as with any programming language; it is because of this that you may find the worst thing to do is try and understand large assembly listings of other authors, or maybe they write very messy programs anyway!

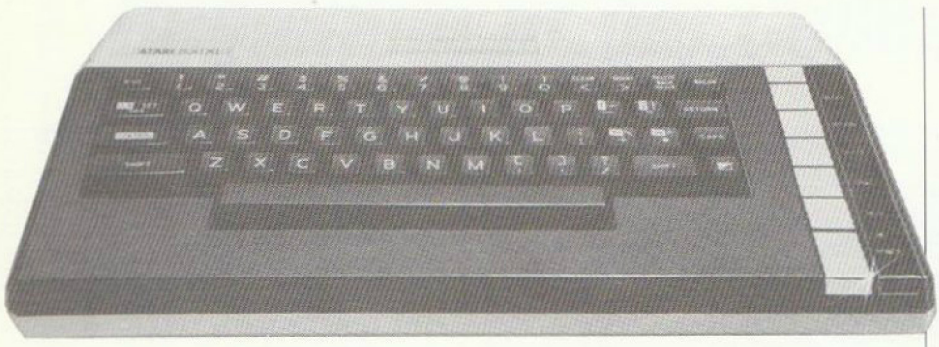
Writing a so-called efficient program can be roughly split into two areas: the first is selecting a good algorithm (method), this is important because it governs the way you will actually structure your program, and equally important to many programs is the way you structure any data you use in the program, these are very fundamental selections and have to be taken before you can start writing your program. If you change your basic algorithm or the way your data is structured, then this will generally mean a large re-write of at least some of your program. The choice of these criteria usually has some relevance to the architecture, or internal layout, of the processor – after all, it is pointless picking a very simple algorithm if it is going to be almost impossible to implement on the 6502 with any degree of efficiency.

Obviously the underlying algorithm and the structure, or layout, of your data will determine the maximum efficiency, but the second area where you can make a program faster and smaller, or some compromise between the two, will be in the actual selection of instructions to perform the job – in some cases this means using some clever 'tricks', but all this really means is understanding how instructions are executed and taking advantage of certain quirks, however, there are not many in the 6502 and most will be fairly obvious as and when you see them.

I have been purposely vague up to now because it is a matter of personal taste and experience which will determine exactly how you go about coding your program, and of course, the approach to two different programs will probably also be totally different. I will now mention just a few rough rules you might employ in your own programs.

Follow the Rules

The 6502 has a simple internal architecture consisting of only a few instructions and registers. The fact that the 6502 has few instructions can actually be a blessing, after all, there is no vast array of complex instructions to remember and fortunately, there are no glaring omissions which will handicap you severely; we shall see later that some of the real power of the 6502 lies with its addressing modes. Although the 6502 is easy to program because of its simplicity, its main limitations lie with the fact that there are only three general purpose registers at our disposal and there is some irregularity in the way they can be combined in different addressing modes (see the tables in part 4, issue 8). In fact, nearly all the common operations you need can only be performed on the accumulator, which means that in general, a value has to be loaded into the accumulator, manipulated and then saved before the next operation can be performed. You will find that the X and Y registers are usually kept for use with the indexed addressing modes, but when they are free, they can be used to either save temporary values or they can be



used, as they often are, as simply a counter. The key thing to remember is that the less memory fetches that are performed the faster the program will execute, that is, use the registers as fully as possible. A few operations can be performed directly on memory locations, without first having to load the value into a register, namely the increment, decrement and shift, rotate operations.

Special attention should always be made to the body of a loop, it is here that a group of instructions can be executed many times, thus any small time saving you can make in this section will improve the overall speed of the loop by a significant amount. It is very rare that you will be able to avoid memory accesses during a loop, so one way of saving time will be to use zero-page locations as this will save one or two cycles on every access (see any 6502 reference book for information on the number of cycles needed for each instruction/addressing mode). You will probably be aware that zero-page locations are quite valuable, especially when BASIC is around, so these have to be chosen carefully, but there use will help make a program faster and smaller.

Lastly, I will mention a few common 'tricks', these just make efficient use of available instructions. Consider the following:

```
LDA  NUM1
CLC
ADC  NUM2
STA  NUM1
LDA  NUM1+1
ADC  NUM2+1
STA  NUM1+1
```

This is the common way of adding two, two-byte numbers together, which a compiler might generate for 'NUM1+NUM2' in some high-level language, where NUM1 is the low byte of the first number and NUM1+1 is the high byte of that number and similarly for NUM2. However, as we saw last time if the second number is a one byte number i.e. NUM2+1 is equal to zero, then the NUM1+1 will either be unchanged or incremented by one, depending on the state of the carry. So, the last three instructions can be replaced by:

```
      BCC  NOINC    ;Increment
      INC  NUM1+1   ;if carry set.
NOINC
```

This is less cumbersome than the first method and uses less bytes of storage for the code, also, for subtraction a similar optimisation can also be made. If you only wish to increment the two-byte number by one, then the following could be used:

```
      INC  NUM1     ;Change low byte.
      BNE  NOINC    ;Change high byte
      INC  NUM1+1   ;if low byte = zero.
NOINC
```

In this example note that as the 'INC' instruction does not change the carry flag, we have to decide when to change the high byte by a method which does not rely on the carry. It turns out that when the high byte needs incrementing when the low byte has changed from 'FF' to '00', so the high byte is changed when the low byte is equal to zero, hence the 'BNE' instruction.

The corresponding code for a decrement by one is slightly different, that is due to the fact we need to test for the low byte changing from '00' to 'FF', in this case we know that if the low byte is equal to zero before it is decremented then the high byte will also be decremented, so it can be coded as follows:

```
      LDA  NUM1     ;Test low byte.
      BNE  SKIP     ;Decrement high byte
      DEC  NUM1+1   ;if low = zero.
SKIP  DEC  NUM1     ;Now do low byte.
```

Here, the load needs to be performed to set the Z flag depending on the value of NUM1; it could, of course, have used either the X or Y register instead. Just this one set of examples shows how you could write better code than you might have first used, and is the sort of omptimisation a compiler normally cannot make, because it allows for the general case of an arbitrary two byte value to add to NUM1.

Other places where you could use similar tricks is in initialising memory values, for instance, if you wanted to set some locations to '01', some others to '00' and some others to 'FF' then you could use the following:

```
      LDX  #1       ;Initial value 1.
      STX  LOC1     ;Save.
      STX  LOC2     ;Save.
      DEX           ;Decrement to 0.
      STX  LOC3     ;Save.
      STX  LOC4     ;Save.
      DEX           ;Decrement to FF.
      STX  LOC5     ;Save.
      STX  LOC6     ;Save.
```

From this you can see that the use of one of the index registers is more useful than the accumulator as it can be incremented or decremented by one, in a single instruction. Apart from the more obvious fact that multiple stores of the same value should be done together (to save re-loading the register), the useful feature is decrementing zero to obtain 'FF' and conversely, incrementing 'FF' to obtain zero. Note that 'FF' is the two's complement of −1 for an eight bit number, so it makes sense to decrement zero to get −1 and incrementing −1 to get 0!

Another saving that is often made is that of the comparison of the value of counter to its end value, so instead of the following:

```
      LDX  #0       ;Start at zero.
LOOP  .
      .             ;Body of loop.
      .
      INX           ;Increment count.
      CPX  #$5B     ;Test for limit.
      BNE  LOOP     ;Go back if not equal.
```

This is usually used:

```
      LDX  #$5B     ;Start at maximum.
LOOP  .
      .             ;Body of loop.
      .
      DEX           ;Decrement count.
      BNE           ;Go back if not zero.
```

The comparison to zero is not needed at the end of the second version of the loop, because the decrement instruction will set the Z flag appropriately.

Many more examples can be found in the same vein, but it would take too much space to mention them all, however, I hope that it has got you thinking that you can improve other code similarly. As a last point, remember that you can always use the stack to save temporary values on, by the 'PHA' and 'PLA' instructions to save and load the accumulator, but do not forget that you should always pull the same number of bytes as you push − otherwise you could be in serious trouble (unless you are up to some other devious trick!). It is therefore good practice not to place the 'PHA' and 'PLA' to far apart, otherwise you might forget which value was on the stack at that time. There are, however, another two instructions which are rarely used, the 'PHP' and 'PLP' which perform a push and pull on the processor status flags register. You might use 'PHP', say, after you have subtracted a number from the accumulator, this will save a copy of all the status flags, you could then continue working with the value in the accumulator and later use 'PLP' to retrieve the status from the subtraction, after which you would branch depending on the state of one of the flags, maybe the carry.
A solution to this sort of problem without using the above instructions could be quite messy. Another use of the push and pull operations is to use 'PHP' followed by 'PLA' to get a copy of the status flags in the accumulator for testing and 'PHA' with

'PLP' to set the status flags to a certain state.

We will now return to the multiplication routine of last time to show how that can be improved.

Improving the Multiplication

If you recall, the routine in the last part of this series calculated the product of two 8-bit numbers by repeated addition to produce a 16-bit result. Just to show improvements can be made to programs, one astute reader pointed out, quite rightly, that there was little point in loading and saving the contents of the accumulator each time around the main loop. The improvement would be to take the load and store instructions out of the loop altogether, then when the loop has finished the accumulator would hold the low part of the result, so it could be saved back into RESULT at the point labeled EXIT and then follow it with the RTS instruction as before. Anyway, the algorithm used of repeated addition is hopelessly inefficient, especially so if the routine was extended to handle 16-bit numbers, due to the large number of iterations (repetitions) of the main loop which would be needed.

A better algorithm can be found by studying the way we normally multiply two decimal numbers together on paper, this involves considering the multiplicand as a whole and then considering the multiplication of each digit of the multiplier, in turn, with the multiplicand; the result is then the addition of these 'partial products':

```
  225  Multiplicand.
  342  Multiplier.
 -----
  450
 9000  Partial products.
67500
 -----
76950  Result = sum of above.
```

The first partial product is 2*225 which is 450, the second is 4*225 but is shifted to the left by one digit i.e. multiplied by 10, to give 9000, the third follows the same pattern, but is shifted left twice. If we apply this same method to binary numbers, then things simplify. Each digit of the multiplier will be either 1 or 0, so if we multiply by 0 the result will also be zero, if it is 1, then the result will just be the multiplicand. Each partial product will either be zero or the multiplicand itself, but rather than take a copy of the multiplicand and then shifting it left by the appropriate number of places, we can successively shift the multiplicand to the left by one each time around our loop. Listing 1 shows how we can code this algorithm.

The program starts by pulling the values of the multiplicand and multiplier off of the stack and storing them in MLTPND and MLTPLR. Next, the accumulator is loaded with zero to initialise the high byte of the multiplicand, location CC hex, and both parts of the result. The X register is loaded with eight, which is the number of bits in the multiplier and hence the number of times we shall repeat the loop. The main loop, starting at MULT, shifts the multiplier right so that the least significant bit is in the carry; if this is zero then we skip the addition of the multiplicand with the branch if carry clear instruction to SKIP. If the digit was a one, then a sixteen bit addition is made between the current value of result and the multiplicand. Before the loop is repeated, we shift the two byte multiplicand to the left by one bit, this is achieved by the ASL and ROL instructions, causing the carry to propagate between the two bytes and a zero to move into the least significant bit of the low byte, thus multiplying the multiplicand, as a whole, by two. The bit count in the X register is then decremented and the loop continues if there is more to do. On each repetition of the loop, the multiplier will be successively shifted to the right, considering each bit in turn, and the multiplicand will be succesively shifted to the left, thus effecting the multiplication by two each time.

```
0100 ;Improved multiplication
0110 ;by partial products.
0120 MLTPND  =      $CB     ;Multiplicand.
0130 MLTPLR  =      $CD     ;Multiplier.
0140 RESULT  =      212     ;Result for BASIC.
0150         *=     $0600   ;Start on page 6.
0160         PLA            ;Discard number of data.
0170         PLA            ;Discard high byte.
0180         PLA            ;Get low byte.
0190         STA    MLTPND  ;Save as multiplicand.
0200         PLA            ;Discard high byte.
0210         PLA            ;Get low byte.
0220         STA    MLTPLR  ;Save as multiplier.
0230         LDA    #0      ;Y holds result low.
0240         STA    MLTPND+1;Zero multiplicand high.
0250         STA    RESULT  ;Zero result low
0260         STA    RESULT+1;and result high.
0270         LDX    #8      ;X holds bit count.
0280 MULT    LSR    MLTPLR  ;Get next bit of multiplier.
0290         BCC    SKIP    ;Skip if zero.
0300         LDA    RESULT  ;Get result low.
0310         CLC            ;and add
0320         ADC    MLTPND  ;multiplicand low.
0330         STA    RESULT  ;Save back.
0340         LDA    RESULT+1;Get result high.
0350         ADC    MLTPND+1;add multiplicand high.
0360         STA    RESULT+1;Save back.
0370 SKIP    ASL    MLTPND  ;Move the multiplicand
0380         ROL    MLTPND+1;one bit to the left.
0390         DEX            ;Decrement bit count.
0400         BNE    MULT    ;Loop if more to do...
0410         RTS            ;It's all yours BASIC!
```

Listing 1.

Unlike the repeated addition method, this program will make exactly eight iterations of the main loop, whatever the two numbers to be mutiplied together are. This saving is significant if you consider that if both programs were extended to handle sixteen bit numbers: the repeated addition method could make over sixty-five thousand iterations of the main loop, this new method would need only sixteen!

We now have an efficient way of calculating the multiplication of two numbers, but again this too can be improved. If we call the multiplicand A and the multiplier B then the result will be A*B, now if we write B down in binary form, labeling each of the bits of B as b0 through to b7 then we get:

B=b7*128+b6*64+...+b2*4+b1*2+b0

Thus, for example, if B has the value five then b2 and b0 will be one and the rest will be zero. We can now re-write this number as follows:

B=b0+2(b1+2(b2+2(b3+2(b4+2(b5+2(b6+2*b7))))))

So the algorithm to work out the value of B goes like this: take bit 7 and multiply it by 2, add bit 6 to the result and then multiply that result by 2, then add bit 5 and multiply that result by 2 etc. This would continue until bit 0 had been added on in which case we would have the value of the binary number B. However, we wanted the value of A*B, so if you follow this

through multiplying by A throughout, you simply end up with an A in front of each 'b' in the last expression.

The algorithm to work out A*B is now: add bit 7 times A to the result and multiply it by 2, then add bit 6 times A and multiply that result by 2 etc., until we have added bit 0 times A. Of course each multiplication of a bit with A will only result in either zero or A itself, so it is really a case of considering whether to add A or not for each bit of B. Listing 2 shows how this algorithm can be implemented; note that this is not the complete listing – it omits the declaration of the three labels MLTPND, MLTPLR & RESULT and the start of the program which saves the values from BASIC in the first two varibles, as in Listing 1.

This routine uses the accumulator to hold the low part of the result and initialises this and the high byte of the result to zero at the start. Looking at the main loop we see that it first multiplies the result by 2, tests the next bit of the multiplier and then adds the multiplicand to the result if necessary. This is the same as the algorithm above, except that it multiplies the initial value of the result by 2, which of course, gives zero, this extra multiplication in fact saves instructions – so it is useful! Listing 3 shows, yet another way of doing this same multiplication which is even more efficient (and the last, I promise!). This algorithm can be deduced in a similar way to the above one; but I will not describe it fully here. This performs the multiplication by adding the multiplicand to the high byte of the result and then dividing the result by 2, using the ROR instructions, note here that two rotates are used, rather than a shift and then a rotate because it also moves any carry from the previous addition down with it.

That's enough of multiplication routines! There are analogous methods for doing division – but they get worse, so I will not start embarking on such a boring passtime! I hope that after seeing four different ways of writing one program you may think about other ways of writing your own programs, or even re–writing other people's. Now we will start looking at those outstanding addressing modes.

```
0230          LDA   #0       ;Zero result low (held in A)
0240          STA   RESULT+1;and high.
0250          LDX   #8       ;Load X with bit count.
0260 MULT     ASL   A        ;Multiply result by 2
0270          ROL   RESULT+1;and high byte.
0280          ASL   MLTPLR   ;Get next MSB bit in carry.
0290          BCC   SKIP     ;Skip if zero.
0300          CLC            ;Else add
0310          ADC   MLTPND   ;multiplicand.
0320          BCC   SKIP     ;Skip if no carry to high.
0330          INC   RESULT+1;Else add one to high.
0340 SKIP     DEX            ;Decrement bit count.
0350          BNE   MULT     ;Loop back if more to do.
0360          STA   RESULT   ;Save low byte of result.
0370          RTS            ;Finished.
```

Listing 2.

```
0230          LDA   #0       ;Zero high byte (in A)
0240          STA   RESULT   ;and low byte of result.
0250          LDX   #8       ;Load bit count.
0260 MULT     LSR   MLTPLR   ;Get next bit of multiplier
0270          BCC   SKIP     ;Skip if zero.
0280          CLC            ;Else add
0290          ADC   MLTPND   ;Multiplicand.
0300 SKIP     ROR   A        ;Divide result by 2
0310          ROR   RESULT   ;and low byte.
0320          DEX            ;Decrement bit count
0330          BNE   MULT     ;and loop if more.
0340          STA   RESULT+1;Save high byte of result.
0350          RTS            ;and say bye-bye!
```

Listing 3.

Accessing Data

So far, we have only been able to use absolute, or fixed, addressing to access data in the computer's memory, this is fine if all you want your computer to do is to keep multiplying numbers together, but for any practical purposes we need a mechanism like that of the array in BASIC. For example, if we wanted to store a number in 256 consecutive locations, starting at some arbitrary address then the only method we have so far is to use 256 store instructions!

We need a method whereby the processor sees a different address for the store instruction each time. A crude method of effecting this is by manipulating the actual operand bytes of the store opcode in the program, this is referred to as 'modification' as we are modifying the actual code of the program; this can be useful in certain rare circumstances, but it is a method which you should try and avoid at all times! The problem with this method is that you will probably get very confused as to the current value stored for the operand, and unless it is reset, it will have a different value the next time the program is run – worse still is that the code could never be moved to a permanent memory, or ROM, because it would be impossible to change the code, so the method wouldn't work! Having decided that modification is not the best method, we will now consider the methods that the 6502 offers us to access data.

Indexing

To access a table of data which is at a fixed place in memory requires adding an offset to the base address of the table. This mode of addressing is referred to as indexed, and not surprisingly, uses one of the two index registers; the assembler accepts the following to indicate this addressing mode:

```
STA  TABLE,X   ;Indexing with X register.
STA  TABLE,Y   ;Indexing with Y register.
```

The instruction is written as normal but has a comma and an index register name following the operand. If, for example, TABLE has the value of $4310, then the machine code will be the following: 9D 10 43 note that no extra information has been stored for this than an absolute access to location $4310, the only difference is the opcode of 9D instead of 8D for an absolute access. The opcode 9D still means store the accumulator but instructs the 6502 to use an index register in its access of the data; note that another opcode is used if we want to use the Y register, in this case it is 99 – see the opcode tables in issue 8. We will now see what happens when one of these new opcodes is executed: The operands are first fetched from memory, in this case it is $10 followed by $43, to form a sixteen bit address, now instead of using this address as it is i.e. an absolute access, the contents of the specified index register is added on first to provide the final computed address which is used for the access, note that the index register is left unchanged. You can think of the instruction as being written as the following if you like:

```
STA  TABLE+X
```

However, this is not used by the assembler to indicate indexed addressing as it might think you were writing a constant expression where X was a label name, so you will have to get used to reading the comma in the instruction as

either 'indexed by' or 'plus'.

The following piece of code demonstrates the use of indexed addressing to store the number $5B into the 256 consecutive locations starting at $4310:

```
        LDA  #$5B     ;Data to store.
        LDX  #0       ;Load index.
LOOP    STA  $4310,X  ;Save using index.
        INX           ;Increment index.
        BNE  LOOP     ;Loop until finished.
```

Before the loop is entered, the accumulator is loaded with the data to be stored and the X register is loaded with the initial index of zero. When the store is first executed, the address will be computed to be $4310 as zero is being added on from X, so $5B is saved into location $4310. The value of X is then incremented by one and a branch is made back to the store, but this time the computed address will be $4311 because one is being added from the contents of X, so the value of $5B will be stored in location $4311. This loop will continue storing the $5Bs into consecutive locations until the value of X returns to zero, hence all the locations from $4310 to $440F will be set to the value $5B by this indexing mechanism. Note that the Y register could have been used instead of X in the above program to exactly the same effect.

Now we have a way of accessing a table, we can write a small program to move a table from one address to another:

```
        LDX  #0          ;Zero index.
COPY    LDA  TABLE1,X    ;Get from
                          first table.
        STA  TABLE2,X    ;Place in second.
        INX              ;Increment
                          common index.
        BNE  COPY        ;Copy rest of table.
```

Assuming TABLE1 and TABLE2 are valid labels holding the addresses of the two tables, then the above program will read each consecutive byte of the the first table and deposit the value in the second table using the accumulator to effect the transfer. Less than 256 bytes could be moved if we either compared X to a limit value before the branch, or by loading X with the limit value and then decremented X down towards zero.

The main limitation with the 6502's indexing is that as the X and Y registers are eight bits wide we can only access tables of 256 bytes or less in length. Another problem we are nearly always faced with is that the items or tables we wish to access keep moving around the memory! We will now see how the remaining addressing modes solve this problem.

Indirecting

The indirect addressing mode of the 6502 allows us to access data via a pointer in memory. For instance, if locations $600 and $601 contain $10 and $43 respectively, then $600 and $601 are said to point 'indirectly' at location $4310, that is to say $600 and $601 hold the address of the location we wish to access. The 6502 only has one instruction which can actually use indirect addressing by itself – it is the jump instruction and is used like this:

```
JMP ($600)
```

Assuming that $CB and $CC still point to location $4310, then this instruction will jump to and try and execute the code at location $4310. Again, no extra information is stored with the jump instruction, instead another opcode is used to indicate that the address specified is to be used indirectly, note that only the first address of the pointer is given; the second byte is always taken from the next consecutive address.

Mixing Indirection with Indexing

All other instructions that can use indirect addressing e.g. LDA or STA, cannot perform indirect addressing by itself, so the following is illegal to the assembler:

```
LDA ($600)
```

Instead of this, the 6502 limits us to page zero for the address of the pointer, and forces us to use indexed addressing with it, which is a combination of the two previous modes. The following show how we write this new mode into the assembler, note that in this case the X register cannot be substituted for Y:

```
LDA ($CB,X)
```

This mode is termed as pre-indexed, as the contents of the X register is first added to the address $CB to give the final address (this will always be in page zero) from which the pointer is taken. So, if X contains $10, then the 6502 will add this to $CB and go to the computed address of $DB; it will then take the contents of $DB and $DC as the pointer to the final location, which could be anywhere in memory. This mode is rarely used, however, as it only allows us to access one item via a pointer, if we wanted to then access the next location in memory, we would have to do a two byte increment on the pointer in memory.

```
0100 ;Call from BASIC with
0110 ;X=USR(1536).
0120         *=     $0600
0130         PLA
0140         LDY    #0
0150 LOOP    TYA
0160         STA    ($58),Y
0170         INY
0180         BNE    LOOP
0190         RTS
```

Listing 4.

The last mode is called post-indexed and can only use the Y register, it is written as follows:

```
LDA ($CB),Y
```

This post-indexed mode takes the contents of a page zero location and uses that indirectly, once it has obtained the location from this indirection, the Y register is added to the address to give the final address to be used. For instance, if $CB contained $10 and $CC contains $43 and Y contains $20, then $CB and $CC will be used to give the address of $4310 and the Y register will be added on to give the final address of $4330.

The use of this indirect-indexed mode allows us to have a series of pointers held in page zero which can then be used to access data tables in memory, indexed by the Y register, however, we still have the limitation that each table cannot be longer than 256 bytes, that is unless we change the value of the pointer. We shall explore the use of these modes in the next issue, but for now consider what Listing 4 does.

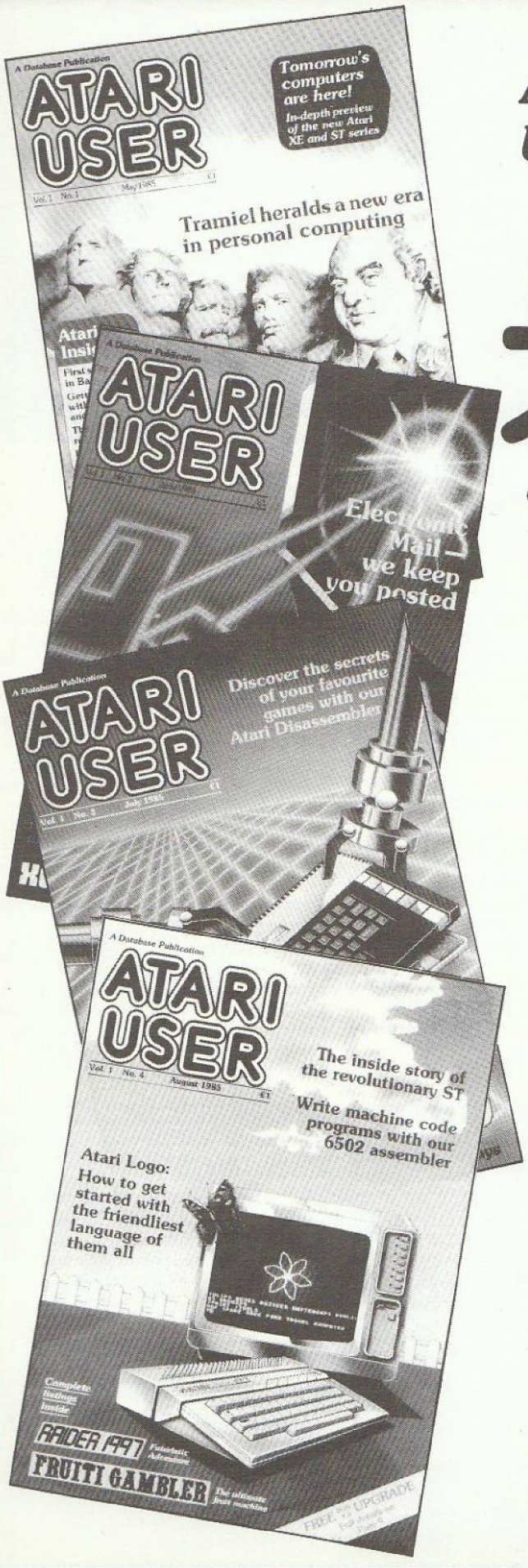
```
XQ 10 DIM FILE$(14)
DA 20 CH=1:? "Please enter device/file na
   me..."
AQ 30 INPUT FILE$
MC 40 GOSUB 10000
YT 50 END
RU 10000 OPEN #CH,4,0,FILE$:TRAP 10160
JY 10010 GET #CH,X:GET #CH,Y
WJ 10020 IF X<>255 OR Y<>255 THEN 10170
QU 10030 GET #CH,SL:GET #CH,SH
NQ 10040 ST=SL+256*SH
VA 10050 GET #CH,EL:GET #CH,EH
TU 10060 EN=EL+256*EH
BA 10070 IOCB=832+CH*16:LN=EN-ST+1
LR 10080 LH=INT(LN/256):LL=LN-LH*256
JO 10090 POKE IOCB+2,7
MC 10100 POKE IOCB+4,SL:POKE IOCB+5,SH
FN 10110 POKE IOCB+8,LL:POKE IOCB+9,LH
JE 10120 X=USR(ADR("hhh*LVd"),CH*16)
QX 10130 GET #CH,SL:GET #CH,SH
YW 10140 IF SL=255 AND SH=255 THEN 10130
YL 10150 GOTO 10040
WA 10160 CLOSE #CH:RETURN
TW 10170 ? "File does not have binary hea
   der.":RETURN
```

Listing 5.

Listing 5 is a BASIC program which will allow you to load any object files you might have created with your assembler, but is mainly intended for cassette users as it will load the object files created by the Assembler/Editor cartridge. For cassette just type 'C:' in response to the prompt, disk users should type 'D:' followed by some filename as usual, however, you will probably find DOS more convenient unless you want to incorporate the routine in one of your programs.

Note that on line 10120 of the BASIC program, the '*' and the 'd' should be typed in inverse video. This is part of the machine code which speeds the loading of the file considerably over that of using BASIC's GET statement in a loop!

Now is your chance to get the assembler out again and do lots of experimenting...



A special offer for User Group members!

£3 OFF

each new subscription

Here's your chance to make sure of a regular copy of Atari User – the independent magazine that's become the premier source of information on the whole range of Atari microcomputers.

Each month brings you a choice selection of first rate programs, powerful utilities, easy to follow articles and invaluable hints and tips – plus full coverage of what's happening in the world of the Atari, with in-depth reviews of all the latest products.

If you want to know all there is to know about your Atari micro, Atari User is essential reading!

ORDER NOW!

Pleases send me the next 12 issues of Atari User at the specially reduced User Group members' price of £9 (normal price £12).
Please indicate method of payment.

☐ Access/Mastercard/Eurocard

☐ Barclaycard/Visa

☐ Cheque/PO made payable to Database Publications Ltd.

Name _____

Address _____

_____

Signed _____

Send to: *Atari User*, FREEPOST, Europa House, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.

(No stamp needed if posted in UK)

MON

DIGITAL PICTURES

by Mark Brighton

Computers of TV and Science Fiction fame are all-powerful, all-knowing devices, capable of controlling anything from a space-ship to a Galactic Empire and its subjects. Bearing this in mind, cast an eye over your own 'home' micro. At first glance it doesn't seem to compare too well. But wait. If you had a high power industrial Laser it would be quite possible to build and fit simple control circuits to aim and fire it using your computer. Add to this a sound locating device, using microphones to pinpoint the direction of any noise in the area, feed this information back to your computer in a digital form and program it to fire the Laser at the noise source. Hey Presto! Your computer becomes as dangerous as a Dalek (and considerably less friendly).

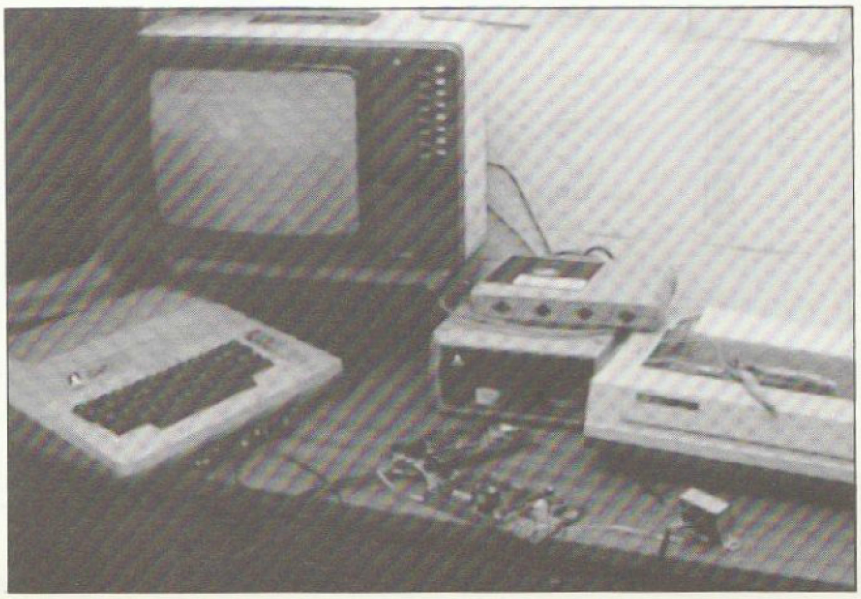
The point I am making is that the type and number of jobs a computer can do in the real world is entirely dependent on how well it can sense and manipulate objects in that world, using its 'peripherals'. These are the devices which interface the computer, which can be thought of as an electronic brain, with the outside world. Consider your micro again. Examine its peripherals: the screen,its keyboard, a loudspeaker, joysticks, a printer, cassette and disk drives, etc. Your computer is well set up for programming and visual display but it is essentially deaf, dumb, blind and paralysed. Small wonder then that it can achieve little in the real world as it stands.

However, one of the exciting things about a computer is that, given access to the busses or I/O ports, almost any kind of circuit can be joined to it to extend its 'senses'. There are quite a few 'add-ons' around which allow sensing and/or control of objects, devices such as Light Pens, Graphic Pads, Mice, etc. which mainly allow interactive 'user friendly' computing to occur. What we are looking for are circuits to give electronic hearing, sight, touch, etc. One such circuit has been published in the December issue of 'Electronics – the Maplin Magazine'. It is a Video Digitiser for use with the older 800/400 machines. A Video Digitiser adds the facility of sight to your computer, using a video camera as the eye.

In simple terms, a Digitiser samples a video signal and converts the picture information into a series of digital numbers representing the brightness of different points within the picture. These numbers are passed to your computer via an input port and stored for processing. The only remaining problem to be overcome, before the computer can make any use of this information, is relating these brightness levels with definate positions within the picture.

In order to understand how this positional information is obtained, it is necessary to take a closer look at the construction of a video signal. Most of you will know that a TV picture is built up line by line, down the screen, with 625 lines (in the U.K. anyway) in a complete picture. For reasons which need not concern us here, a complete picture consists of two separate frames, which is to say that the flying spot of light produced by the electron gun at the back of the TV tube, traces two sets of lines from the top to the bottom of the screen for each whole picture.

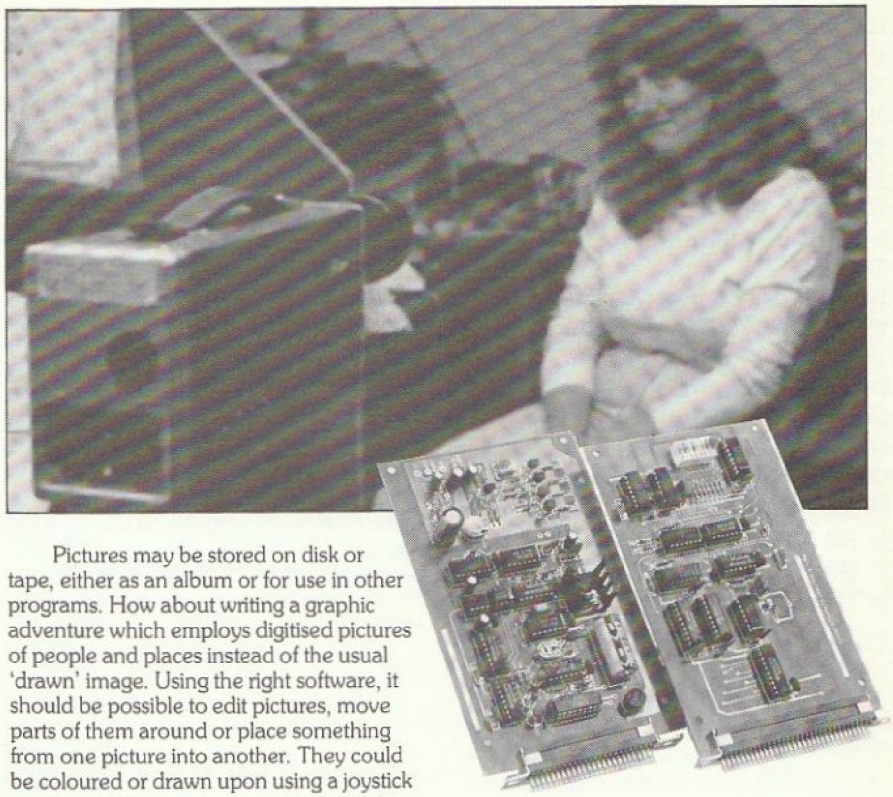
There are 312.5 lines in each frame and the end of each frame is separated from the beginning of the next by a special signal called a Frame Synchronisation Pulse. Likewise, the lines are separated by Line Synchronisation Pulses. So, by using these signals, the satrt of each frame and each line can be found. The computer counts the Line Synch Pulses since the last Frame Synch Pulse to determine the vertical position of the current sample within the picture. It tells the digitiser where along the line to take a sample, by sending a digital number corresponding to the left/right position, to a circuit in the digitiser that waits for a Line Synch Pulse, and then delays the signal controlling the sampling for a period determined by the position number. Obviously a short delay gives a sample on the left and a longer delay gives samples further to the right. In this way, the computer can gain access to brightness information from anywhere within the



frame, and keep track of where it belongs.

The actual conversion of the video signal to digital numbers is handled by an analogue to digital converter chip such as the Ferranti ZN427. This chip compares the video input with a number of internal reference levels to derive an eight bit digital output, taking as little as 10us (10 millionths of a second) to do so.

Having stored the picture, what can be done with it? First and formost, it can be displayed on the screen, good results being obtained on screens with a high X,Y resolution and many grey levels. It may be necessary to turn down the colour on your TV/Monitor in order to obtain enough grey levels when using graphics modes without luminance control. Some experimentation with different subjects and various angles of lighting will soon familiarise you with the conditions required to get the best pictures. Digitisation is not, for example, suitable for use with fast moving subjects since the conversion of a complete picture takes around 5 seconds, even using machine code software. An amusing by-product of this limitation is to move slowly across the picture area whilst the conversion is taking place. After a little practice, pictures of bizarre, wedge shaped heads or impossibly long curly fingers can be obtained.



Pictures may be stored on disk or tape, either as an album or for use in other programs. How about writing a graphic adventure which employs digitised pictures of people and places instead of the usual 'drawn' image. Using the right software, it should be possible to edit pictures, move parts of them around or place something from one picture into another. They could be coloured or drawn upon using a joystick

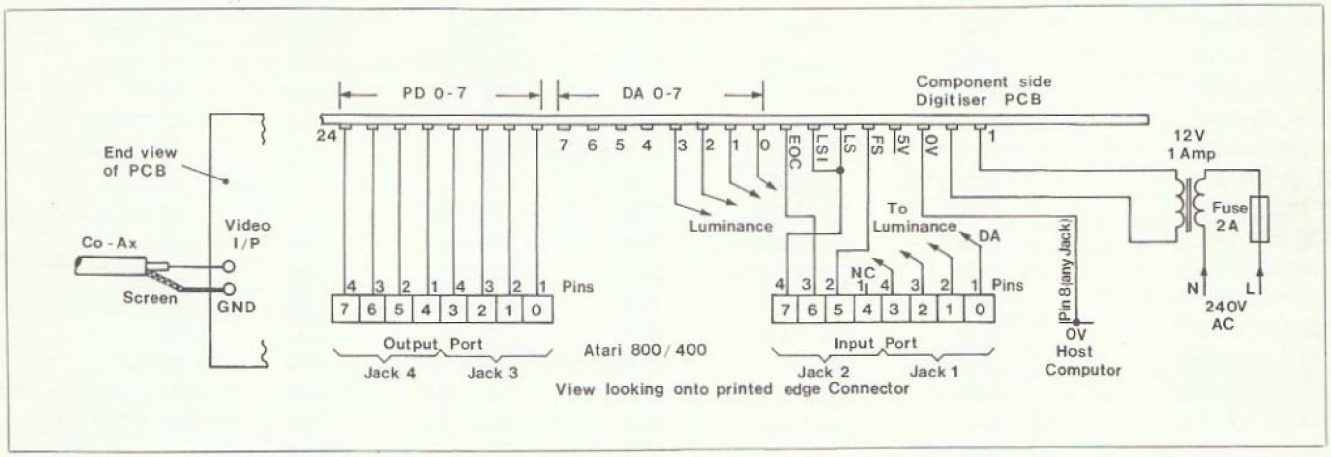


Figure 1. Single Board System.

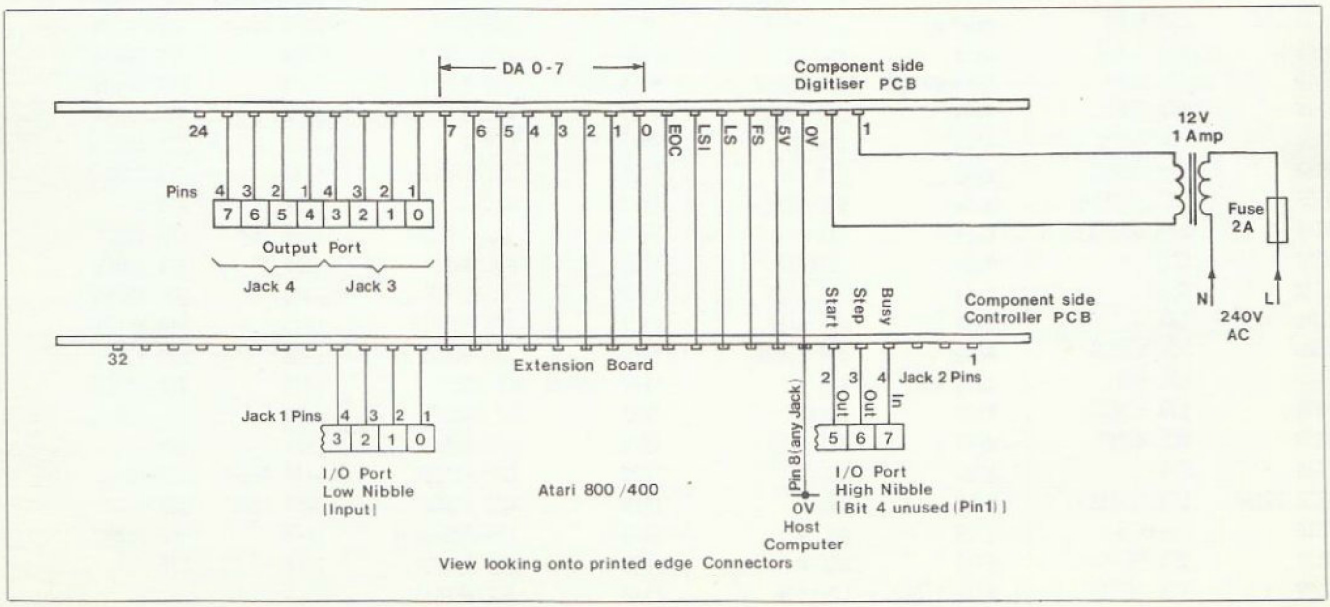


Figure 2. Two Board System.

```
LC 5 GRAPHICS 9
KH 10 REM .*   SET UP PORTS     *
TV 20 POKE 54019,56
ZE 30 POKE 54017,255
QU 40 POKE 54019,60
TL 50 POKE 54018,56
TU 60 POKE 54016,112
QK 70 POKE 54018,60
QR 80 C=0
CV 100 PA=54016:PB=54017
BJ 200 POKE PB,230
YI 250 POKE PA,64:POKE PA,0
LK 300 IF PEEK(PA)/128>=1 THEN GOTO 300
JA 400 FOR X=0 TO 191:LUM=PEEK(PA)
LY 410 GOSUB 2000:POKE PA,32
KM 420 POKE PA,0:NEXT X
QB 430 C=C+1:POKE PB,230-C
NX 500 GOTO 250
OY 2000 IF C=80 THEN GOTO 2000
GH 2010 COLOR LUM:PLOT C,X:RETURN
```

Listing 1.

or light pen. Other possibilities involve the application of artificial intelligence programs to examine scenes and recognise objects or detect changes in the pictures. As an intruder alarm, the computer could perhaps recognise the difference between a burglar and the family cat, avoiding false triggering of the alarm. Systems similar to this are used in industry, to recognise objects on a conveyor belt and sort them into the correct bins.

The complete circuit details for the Digitiser are shown in the 'Electronics' magazine article together with a full parts list and wiring details. A controller board is also available, which takes the hard work out of the programming. The hardware on this board detects and counts line synch pulses, sets the vertical start position and stores a whole column of picture data in a buffer RAM, to be read at your convenience. Using the controller it is possible to control the Digitiser from Basic. The Digitiser can be used on its own, if you feel your machine code knowledge is up to it, and wiring the board up to the joystick ports is shown in Figure 1. Wiring up the two board system is shown in Figure 2. The Digitiser is sold in kit form for £41.95 and the controller costs £29.95, but note that this does not include any cables or connectors for fixing up to the computer, these you have to obtain yourself.

Listing 1 is a Basic program for use with the two board system which simply displays the picture information on a Graphics 9 screen, it will need adding to in order to save the pictures to disk or cassette. Listing 2 does the same job as the Basic program but it only takes 5 seconds to display a complete picture against several minutes for the Basic program.

This article has only looked at a few of the potential uses of a Digitiser. The users imagination and programming ability being the real limits to its applications. If you do decide to extend your computers senses by adding a Digitiser, I am sure you will find it a great challenge and exceedingly interesting to boot!

Listing 2.

```
10   *=$7000
20 PORTA=$D300
30 PORTB=$D301
40 PACTL=$D302
50 PBCTL=$D303
60 NMIEN=$D40E
70 IRQEN=$D20E
80 POKMSK=$0010
90 DMACTL=$D400
0100 SDMCTL=$022F
0110 SAVMC=$0058
0120 MASK=$06F9
0130 LINES=$06FA
0140 POSIT=$06FB
0150 SCNCOL=$06FC
0160 OFFSET=$00D4
0170 FINCOL=$06FE
0180          PLA
0190 INIT     LDA #$00
0200          STA LINES
0210          LDA #240
0220          STA POSIT
0230          LDA SAVMC
0240          STA SCNCOL
0250          LDA SAVMC+1
0260          STA SCNCOL+1
0270          CLC
0280          CLD
0290          TAX
0300          LDA SCNCOL
0310          ADC #40
0320          STA FINCOL
0330          BCC NOINT
0340          INX
0350 NOINT    STX FINCOL+1
0360          LDA#$38
0370          STA PBCTL
0380          STA PACTL
0390          LDA#96
0400          STA PORTA
0410          LDA#$FF
0420          STA PORTB
0430          LDA#$3C
0440          STA PBCTL
0450          STA PACTL
0460 START    LDA SCNCOL
0470          STA OFFSET
0480          LDA SCNCOL+1
0490          STA OFFSET+1
0500          LDA POSIT
0510          STA PORTB
0520 TOG      LDA#64
0530          STA PORTA
0540          JSR DELAY
0580          LDA#0
0590          STA PORTA
0600 HI       BIT PORTA
0610          BMI HI
0620 GETLUM   LDX PORTA
0630          LDA POSIT
0640          LSR A
0645          LSR A
0650          BCC MULT
0660          TXA
0662          LDX#240
0664          STX MASK
0670          AND #$0F
0680          JMP STORE
0690 MULT     TXA
0692          LDX#15
0694          STX MASK
0700          ASL A
0710          ASL A
0720          ASL A
0730          ASL A
0740 STORE    LDY#$00
0750          ORA (OFFSET),Y
0760          STA (OFFSET),Y
0762          LDY#40
0764          LDA(OFFSET),Y
0765          AND MASK
0766          STA(OFFSET),Y
0770 NEWROW   CLC
0780          CLD
0790          LDA OFFSET
0800          ADC #40
0810          STA OFFSET
0820          BCC LNCHK
0830          CLC
0840          INC OFFSET+1
0850 LNCHK    INC LINES
0860          LDA LINES
0870          CMP #191
0880          BCC STEP
0890          LDA #$00
0900          STA LINES
0910 ODD      LDA POSIT
0920          LSR A
0925          LSR A
0930          BCC PICFIN
0940          CLC
0950          LDA SCNCOL
0960          ADC #$01
0970          STA SCNCOL
0980          BCC PICFIN
0990          INC SCNCOL+1
1000 PICFIN   DEC POSIT
1005          DEC POSIT
1010          LDA SCNCOL
1020          CMP FINCOL
1030          BCC NEWCOL
1040          LDA SCNCOL+1
1050          CMP FINCOL+1
1060          BCC NEWCOL
1070 FIN      RTS
1080 STEP     LDA#32
1090          STA PORTA
1100          JSR DELAY
1150          LDA#0
1160          STA PORTA
1170          JMP GETLUM
1180 NEWPIC   LDA#$16
1190          STA POSIT
1200          LDA SAVMC+1
1210          STA SCNCOL+1
1220          LDA SAVMC
1230          CLC
1240          ADC #40
1250          STA SCNCOL
1260          BCC NEWCOL
1270          INC SCNCOL+1
1280 NEWCOL    JMP START
1290 DIS       LDA #$00
1300          STA IRQEN
1310          STA POKMSK
1320          STA NMIEN
1330          STA DMACTL
1340          STA SDMCTL
1350          RTS
1360 ENA      LDA #$C0
1370          STA IRQEN
1380          STA POKMSK
1390          STA NMIEN
1400          LDA #34
1410          STA DMACTL
1420          STA SDMCTL
1430          RTS
1440 DELAY    LDX#30
1450 DOWN     DEX
1460          BNE DOWN
1470          RTS
```

DISK JACKET

by Alan Goldsbro - Leeds

This program came about because I always seem to loose my disk jacket covers. I've either left them at friends houses, never to be found again or I've covered the jacket in scribbly writing and having reformated the disk and now whats on the jacket is not what is on the disk. So I promptly wrote this program which prints out the shape of a disk jacket on my Epson printer. All I have to do then is cut out the shape, fold it around a disk and stick the sides together with some glue, and I have a brand new cover.

Type in the program using KEYO to check your typing (or you can use TYPO), then save off once the checksum program has confirmed you have made no errors. If you are not using KEYO, ignore the two code letters before each line number.

```
MO 10 REM ************************
RU 30 REM * CUSTOMISED DISK JACKET *
FL 40 REM *                      *
LG 50 REM *     ALAN GOLDSBRO    *
FN 60 REM *                      *
RA 70 REM *     MONITOR  1985    *
MW 90 REM ************************
XO 110 POKE 752,1:REM TURN OFF CURSOR
RZ 120 POKE 710,148:POKE 712,148:REM SELE
   CT SCREEN/BORDER COLOR
PB 125 DIM CLS$(1):CLS$=CHR$(125)
OQ 130 ? CLS$:POSITION 12,0:? "***DISK JA
   CKET***"
JN 140 POSITION 2,2:? "This program is de
   signed to print out a customised Disk
   Jacket, there are no"
IK 150 POSITION 2,4:? "user defined param
   eters, all the      workings are in th
   e listing."
XC 160 POSITION 2,8:? "You will need to s
   et up your printer  (Epson) with the c
   olor paper of your"
KC 170 POSITION 2,10:? "Choice, (A4 size)
   , it will need to be at least 100gms w
   eight."
WJ 180 POSITION 2,14:? "The printer will
   mark out the corners of the Jacket wit
   h a '+', you then"
DF 190 POSITION 2,16:? "join up the marks
    and cut along the   lines."
RO 200 POSITION 9,22:? "PRESS START TO CO
   NTINUE"
WJ 210 IF PEEK(53279)<>6 THEN 200:REM CHE
   CK FOR START
UA 220 ? CLS$:POSITION 2,2:? "There are s
   paces marked out for labels(89x36mm) w
   ith your programs listed onthem."
CE 230 POSITION 2,6:? "A number of public
    domain programs arearound which are c
   apable of listing"
RM 240 POSITION 2,8:? "out a directory in
    condensed print."
TW 250 POSITION 2,10:? "When you have cut
    out the jacket, the sides will need t
   o be firmly stuck"
OW 260 POSITION 2,12:? "with a suitable a
   dhesive and left to  dry for a short w
   hile."
YV 270 POSITION 2,16:? "Place paper in pr
   inter and press      SELECT when ready
   ."
AR 280 IF PEEK(53279)<>5 THEN 270:REM CHE
   CK FOR SELECT
NX 290 ? CLS$:POKE 752,1:POSITION 10,10:?
    "Setting up Printer"
UD 300 OPEN #1,8,0,"P:":PRINT #1;CHR$(27)
   ;CHR$(64):REM Initialise The Printer.
QP 310 POSITION 9,20:? "PRESS START TO PR
   INT"
UE 320 IF PEEK(53279)<>6 THEN 310
CA 330 ? CLS$:POKE 710,48:POKE 712,48
KH 340 POSITION 10,10:? "Printing Disk Ja
   cket"
CL 350 PRINT #1;"  +----------------------
   ------------------------------------
   ----------+":REM TOP LINE
JA 360 PRINT #1;CHR$(27);"G";CHR$(14);"
              YOUR NAME":REM Double Stri
   ke & Double Width(ONE LINE ONLY)
RI 370 PRINT #1;"                      *****
   *********************************"
SM 380 PRINT #1;"                       *
                                       *"
SO 390 PRINT #1;"                       *
                                       *"
RX 400 PRINT #1;"                       *
                                       *"
CP 410 PRINT #1;"                       *
         Place Label Here              *"
SB 420 PRINT #1;"                       *
                                       *"
SD 430 PRINT #1;"                       *
                                       *"
SF 440 PRINT #1;"                       *
                                       *"
SH 450 PRINT #1;"                       *
                                       *"
RH 460 PRINT #1;"                        *****
   *********************************"
RJ 470 PRINT #1;"                        *****
   *********************************"
SN 480 PRINT #1;"                       *
                                       *"
SP 490 PRINT #1;"                       *
                                       *"
RY 500 PRINT #1;"                       *
                                       *"
CQ 510 PRINT #1;"                       *
         Place Label Here              *"
SC 520 PRINT #1;"                       *
                                       *"
SE 530 PRINT #1;"                       *
                                       *"
SG 540 PRINT #1;"                       *
                                       *"
SI 550 PRINT #1;"                       *
                                       *"
CW 560 PRINT #1;"  +                     *****
   *********************************
              +":REM CORNER FLAPS
ZV 570 PRINT #1;CHR$(27);"H";:REM Turn Of
   f Double Strike.
SP 580 PRINT #1:PRINT #1:PRINT #1;"
     +
                        +":REM MIDDLE
KO 590 FOR I=1 TO 27:PRINT #1:NEXT I
JT 600 PRINT #1;CHR$(11);"         +-----
   ------------------------------------
   -------------+":REM Activate Printer.
OY 610 PRINT #1;CHR$(24);CHR$(27);"@";:RE
   M Carriage Return.
BR 620 ? CLS$:POKE 710,162:POKE 712,162
SM 630 CLOSE #1:POSITION 2,4:? " Disk Jac
   ket finished!"
JL 640 POSITION 2,8:? " Press SELECT to p
   rint again"
DV 650 POSITION 2,12:? " Press START to e
   nd."
LT 660 IF PEEK(53279)=5 THEN GOTO 290:REM
    START AGAIN
YB 670 IF PEEK(53279)=6 THEN END
MV 680 IF PEEK(53279)<>6 OR PEEK(53279)<>
   5 THEN 660:REM CHECK FOR START OR SELE
   CT
```

OPENING OUT

AN INTRODUCTION TO THE USE OF FILES Part 2

By Ron Levy

In the first part of Opening Out I explained the principle of the Atari's device – independent system, and demonstrated how to create small files using the PRINT command. In this issue I will show you how you can use your cassette or disk drive to create your own database.

For this example we will use what is perhaps the most common type of database – a name and address file. Let us suppose that we want our file to store, for each entry, 5 lines of name & address, with 25 characters per line, and 3 additional lines of 25 characters for any other information you might decide necessary, for each entry. Ultimately we will want this information stored permanently on disk or cassette, laid out in some kind of order so that we could quickly call up, examine, edit or replace any entry quickly and efficently. This is of course not a problem with a disk drive, since disk drives have the ability to perform "RANDOM ACCESS" i.e. the read/write head can move instantly to any part of the file and pick out a particular entry.

For those of you who own disk drives, you will be able to use this method, but in the next part of this series I will show you how to take advantage of the more specialised disk I/O functions and give an updated version with true random disk access.

So Let Us Begin

Our first task is to decide how we can store our addresses in memory. We have decided that we want to store 8 lines of 25 characters, which gives a total of 200 characters, for each address. It is of course impractical to have a separate string variable for each line, so we need to use one very large string and divide it into many separate sections, or sub-strings. In the program I have called our large string N$, and Diagram 1 shows how this is done a little more clearly.

We will be using N$ as the main storage string so this will be many thousands of characters long, but to make programming simpler we will use another string called T$ to temporarily hold an address to be edited or displayed. This will of course be DIMensioned to just 200 characters.

Suppose that we want to extract address number 3 from our main store. We must first calculate the position of the first character of address 3 inside N$, and since there are 200 characters per address this will be:–

```
(3-1)*200+1
```

Don't forget that characters inside a string begin with number one and not zero, i.e. N$(1,1) is the very first character in N$, there is no such thing as N$(0,0). We can generalise our above equation by saying that for address N, the first character in N$ is:

```
(N-1)*200+1
```

Since the last character in N$ for address N will be 200 on from this, its calculation is simply:

```
N*200
```

Therefore, to extract (and place into T$) the data for address N we write:

```
T$=N$((N-1)*200+1,N*200)
```

Once we have extracted our address in T$ we can more easily examine and alter specific lines of the address. So how do we separate the lines from each other? We use the same principles as we used to obtain T$, except that we have 25 characters per line. This is how our lines are made up in T$:–

```
Line 1 = T$(1,25)
Line 2 = T$(26,50)
Line 3 = T$(51,75)
Line 4 = T$(76,100)
Line 5 = T$(101,125)
Line 6 = T$(126,150)
Line 7 = T$(151,175)
Line 8 = T$(176,200)
```

Suppose that we want to make L$ equal to the contents of line 5, we would say:–

```
L$=T$(101,125)
```

Just as we did before with N$, we can also generalise the equation for line X of T$ thus:

```
L$=T$((X-1)*25+1,X*25)
```

What About Storing Our Data?

With Atari BASIC you can print a string of any length to a file, and so this is the method I have used in our program. We could of course have used the PUT command to save the contents of N$ thus:

```
FOR X=1 TO LEN(N$)
PUT#1,ASCII(N$(X))
NEXT X
```

Although this would work it would in fact take much longer to run, as well as being a longer and more complex routine. When we use PRINT#1,N$, the operating system performs the same function in machine code resulting in a much faster transfer.

There is something else I have used in our program which speeds up the save routine on cassette drives. Take a look at line 3100, where the cassette is opened as a file the open command is:

```
OPEN #1,8,128,FILE$
```

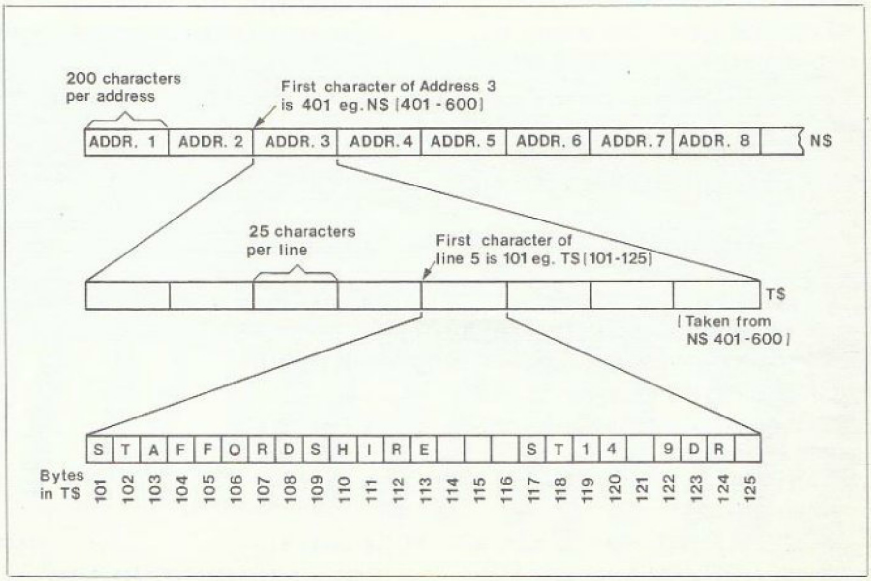


Diagram 1.

The 8 signifies (as it does also with disk files) that we are opening the file for output to the cassette, or WRITE MODE as it is often called. The 128, however, is a special mode for the cassette handler, it forces the handler to use short INTER-RECORD GAPS. IRG's as they are called, are the spaces between the bursts of noise you can hear when data is being sent to the cassette. When you save a program to cassette using the CSAVE command, BASIC also uses short inter-record gaps, but when you LIST"C:" or CSAVE it uses long IRG's. The use of long IRG's enables the cassette to stop if necessary between bursts of data, something which we would need to allow for if we were using the PUT #1 command instead of PRINT #1.

Re-Loading Our File

Although we are able to use the PRINT #1 command to save the information in N$, unfortunately it is not possible to use the INPUT #1 command to retrieve it. Using just BASIC, this would only leave one other method, the GET #1 command in the following loop:

```
FOR X=1 TO 100000
GET#1,Y:N$(X)=CHR$(Y)
NEXT X
```

Notice that I have used a large dummy value of 100,000 in the FOR-NEXT loop. This would never be reached because BASIC would give an END OF FILE error which the program would have to TRAP to a CLOSE #1 statement. This loop would, however, prove very slow, especially considering that we would need to use long IRG's for both the save and load routine.

Consequently, in the program I have taken a short cut to the operating system's CIO utility (or CENTRAL INPUT/OUTPUT routine, as you will know if you read Part 1 in Issue 8). Using POKES, I have set up the Input/Output Control Block (IOCB for short) of channel 2. You may remember from the first part of this series that there are eight IOCB's, each 16 bytes long, and that they act as note pads onto which a program can write instructions on the I/O operation to be performed. BASIC does all this for you when you perform instructions such as PRINT #1 or GET #1, but since we want to go directly to CIO we must set this up ourselves.

Notice that although the program is going to load the data file using a direct call, it still OPENs the file using the BASIC command since this is not such a time-consuming operation. Once the file has been OPENed we can begin setting up IOCB 1, and the first task is to install the command byte. This tells CIO what operation we actually want to perform, and on line 21050 you will see that to do this I have POKEed location 850 with the number seven, this being the command to get a series of bytes. Next we must tell CIO where to put the data, and if you look at line 21070 you will see that we are POKEing two numbers into locations 852 and 853. The values to poke into these locations are derived from the command

ADR(N$), and they represent the starting memory location of N$, the string we will be using as our input buffer. Memory locations 852 and 853 are called the buffer pointers, and you will often find them refered to by the names ICPTL and ICPTH. They are the low and high bytes respectively of the 16 bit binary address where we want our data to begin loading.

On line 21080 you will see POKE 857,255. Here I have taken something of a short-cut, because the locations 856 and 857 are used to store the least significant byte (LSB) and the most significant byte (MSB) respectively of the number of bytes of data we want CIO to load from our file. What I have done is to ensure that they contain an exceedingly large number, since we do not neccesarily know how much data there is in our file. CIO will then continue load until there is no more data left in the file, when CIO will then return an error status in byte 851. This will be error number 136, which means we have tried to read past the end of the file. You can see that our program ensures that this is the reason for the error by PEEKing location 851 in line 21100.

Although we POKE into locations 856 and 857 to tell CIO how many bytes to transfer, when it has completed its operation CIO will change these to tell us exactly how many bytes were really transfered. Obviously, this will only be changed if, due to an error, it could not load as many as we requested (as will be the case in our program). Line 21110 is where the true number of loaded bytes is calculated.

Notice how we call (or 'jump' into) CIO. We use the BASIC command USR, since this is the command which forces the computer's processor to continue running at an address which we can specify. For example:

```
X=USR(1536)
```

This will force the processor to go to address 1536. We would only issue this command if we knew that a machine code program began at that location, because if there wasn't we would see some rather unexpected and unpredictable results – the computer may even lock-up and need switching off and back on again (a cold-start as it is called) to get it working!

In the program, I have placed a machine code program into a BASIC string (and called it CIO$), for convenience. The only drawback, however, is that this string could reside almost anywhere in memory, so I have had to include the command ADR(CIO$) inside the USR instruction. With the USR command there is also the facility to transfer numbers to your machine code routine, and I have used this to pass the number 16 through. This figure represents the number of the I/O channel (and consequently the IOCB's) number that I want CIO to work on. This number is the channel number multiplied by sixteen, so that if I had OPENed our file on channel 2 instead I would have to pass the number 32, and so on.

So why, I here you ask, do we call our own machine code routine and not the CIO entry into the operating system? The answer is, I am afraid, a little complex to grasp unless you are reasonably proficient at machine code programming (in which case you will doubtless already know the answer!), but suffice it to say that we need to do a quick shuffle of the processor's internal storage registers before the routine jumps into CIO. For those of you who are interested the short program held in CIO$ is as follows:

```
PLA
PLA
PLA
TAX
JMP $E456
```

Listing 1.

```
EI 1 REM ********************************
UJ 2 REM **         ---FILER---         **
CM 3 REM ** A database program for      **
NH 4 REM ** use with Disk or Cassette   **
AC 5 REM **    For MONITOR Magazine.    **
DU 6 REM **        By Ron Levy          **
EO 7 REM ********************************
NN 8 REM
FX 100 DIM CLS$(1),CL$(2),T$(200),L$(25),
   DOT1$(1),DOT$(200),XX$(5),FILE$(15),CI
   O$(7)
MZ 120 CLS$=CHR$(125):CL$=CHR$(156):CL$(2
   )=CHR$(157):DOT1$=CHR$(20)
VJ 130 GOSUB 32000:REM Load String Search
TV 140 TAB=85
XF 150 RESTORE 160:FOR XX=1 TO 7:READ YY:
   CIO$(XX)=CHR$(YY):NEXT XX
JM 160 DATA 104,104,104,170,76,86,228
QI 210 FOR XX=1 TO 200:DOT$(XX)=DOT1$:NEX
   T XX
OZ 220 XX=FRE(0)-200:XX=INT(XX/200)
VZ 230 DIM N$(XX*200)
OD 240 PRINT "Maximum number or records="
   ;XX:PRINT
TX 300 REM *** Load A File Into RAM ****
LM 310 REM *** ******************** ****
TE 320 ? "<L>oad or create <N>ew file ";
WG 330 INPUT XX$:XX$=XX$(1,1)
RL 340 IF XX$="N" THEN N$="":N=0:GOTO 500
AR 350 IF XX$<>"L" THEN 320
WP 360 ? "Filename..<C: For cassette>";
AF 370 INPUT FILE$:IF FILE$="" THEN 320
AJ 380 GOSUB 21000:REM Load the file.
SF 498 REM
SI 499 REM
ES 500 REM ******** Main Menu.**********
JV 501 REM ******** ********* **********
UD 510 PRINT CLS$;:POSITION 7,0
RQ 520 PRINT "Name & Address Database."
MZ 530 POSITION 14,2:? TOTAL;" Entries"
TO 540 PRINT
VU 600 ? "    (1) Add a New Entry...(1)"
WH 610 ? "    (2) Edit or Examine...(2)"
VB 620 ? "    (3) Store The File....(3)"
IB 630 ? "    (4) Find an Entry.....(4)"
CS 700 POSITION 0,20:? CL$;:POKE TAB,13
IB 710 PRINT "Option...";
OU 720 TRAP 700:INPUT XX:TRAP 40000
EX 730 XX=ABS(INT(XX))
DZ 740 IF XX<1 OR XX>4 THEN 700
SG 800 GOTO XX*1000
SK 998 REM
SN 999 REM
LU 1000 REM ** Add A Name And Address.**
LT 1001 REM ** ********************** **
JU 1010 PRINT CLS$;:POSITION 10,0
PV 1020 ? "Add A NEW Address."
VK 1030 N=TOTAL+1
VV 1100 T$=DOT$:GOSUB 20000
GI 1110 FOR XX=1 TO 8
EM 1120 POSITION 12,2+XX+(XX>5):INPUT L$
LY 1130 FOR YY=1 TO LEN(L$)
KY 1140 IF L$(YY,YY)=DOT1$ THEN L$(YY,YY)
   =" "
BC 1150 NEXT YY
HF 1160 IF YY<25 THEN FOR YY=YY TO 25:L$(
   YY)=" ":NEXT YY
OA 1170 YY=(XX-1)*25+1:T$(YY,YY+24)=L$
AO 1180 NEXT XX
TT 1200 GOTO 2100:REM EDIT the entry.
KV 1998 REM
KZ 1999 REM
IH 2000 REM ******* Examine/Edit *******
WB 2001 REM ******* ************ *******
GS 2010 NW=0:PRINT CLS$;:POSITION 7,0
VB 2020 PRINT "Examine/Edit An Address."
BZ 2030 POSITION 10,1
IF 2040 PRINT "Address no....";
KL 2050 INPUT XX$:IF XX$="" THEN 500
LN 2070 TRAP 2000:N=VAL(XX$):TRAP 40000
JS 2080 IF N<1 OR N>TOTAL THEN 2000
TK 2090 T$=N$((N-1)*200+1,N*200)
AM 2100 GOSUB 20000
NV 2110 POSITION 10,14
ZG 2120 ? CL$;"Correct <Y/E/1....8> ";
GW 2130 INPUT XX$
QI 2140 IF XX$="E" THEN ? "NOT SAVED ":PR
   INT "  Press Return to continue":INPUT
    XX$:GOTO 500
AQ 2150 IF XX$="Y" THEN 2400
TH 2160 TRAP 2110:XX=VAL(XX$):TRAP 40000
JA 2170 XX=INT(XX):IF XX<1 OR XX>8 THEN 2
   110
ZC 2180 POSITION 12,XX+2+(XX>5):INPUT L$
SW 2190 IF LEN(L$)<25 THEN FOR YY=LEN(L$)
   +1 TO 25:L$(YY)=" ":NEXT YY
NI 2200 YY=(XX-1)*25+1:T$(YY,YY+24)=L$
NT 2210 GOTO 2100
BB 2400 YY=(N-1)*200+1:N$(YY,YY+199)=T$
NL 2410 IF N>TOTAL THEN TOTAL=N
PU 2490 GOTO 500
KW 2998 REM
LA 2999 REM
ZH 3000 REM ***** Save RAM to File *****
MU 3001 REM ***** *************** *****
VV 3010 PRINT CLS$:POSITION 15,0
TX 3020 PRINT "Save to Cassette or Disk"
AX 3030 PRINT :PRINT
CU 3040 ? "Filename..<C: for cassette>";
VO 3050 INPUT FILE$:IF FILE$="" THEN 3000
JC 3100 IF FILE$(1,1)="C" THEN OPEN #1,8,
   128,"C:":GOTO 3200
ZC 3120 OPEN #1,8,0,FILE$
DI 3200 PRINT #1;N$
ZJ 3300 CLOSE #1:SOUND 0,0,0,0
OW 3500 GOTO 500
KX 3998 REM
LB 3999 REM
IJ 4000 REM ****** Find an Entry *******
IY 4001 REM ****** ************* *******
PH 4010 PRINT CLS$;:POSITION 15,0
WM 4020 PRINT "Find an Entry."
NK 4030 POSITION 0,13:PRINT CL$;
DU 4040 ? "You may enter any part of the"
ID 4050 ? "address you wish to find....."
RS 4060 ? :? "...";:INPUT L$
CL 4070 IF L$="" THEN 500
AI 4080 XX=USR(ADR(S$),ADR(N$),LEN(N$),AD
   R(L$),LEN(L$))
XU 4100 IF XX=0 THEN POSITION 0,19:PRINT
   CL$,"No Match Found...";:INPUT XX$:GOT
   O 4000
PP 4110 N=INT((XX-1)/200)+1
ST 4120 T$=N$((N-1)*200+1,N*200)
JQ 4130 GOSUB 20000:REM Display Entry N
YY 4140 POSITION 0,16:PRINT CL$,
BA 4150 PRINT "This one <Y or RETURN>";
IS 4160 INPUT XX$:IF LEN(XX$) THEN IF XX$
   (1,1)="Y" THEN PRINT CLS$:GOTO 2100
KJ 4170 XX=USR(ADR(S$)+71)
FL 4190 GOTO 4100:REM Go for next match.
KY 4998 REM
LC 4999 REM
FA 20000 REM ***** Display Address *****
FA 20001 REM ***** *************** *****
WI 20010 POSITION 2,3:FOR XX=1 TO 9:? CHR
   $(156);:NEXT XX:FOR XX=1 TO 9:? CHR$(1
   57);:NEXT XX
AS 20020 ? "   Name: 1)";T$(1,25)
JK 20030 ? "   Addr: 2)";T$(26,50)
QC 20040 ? "   Addr: 3)";T$(51,75)
QL 20050 ? "   Town: 4)";T$(76,100)
RN 20060 ? " County: 5)";T$(101,125)
BF 20070 ?
BL 20080 ? "         6)";T$(126,150)
IP 20090 ? "         7)";T$(151,175)
CE 20100 ? "         8)";T$(176,200)
UA 20110 POSITION 0,21
YC 20120 PRINT CL$;"Entry Number ";N
QU 20130 PRINT CL$;"***************"
EL 20190 RETURN
EE 20198 REM
EJ 20199 REM
TW 21000 REM ******* Load a File *******
MY 21001 REM ******* *********** *******
DA 21010 IF FILE$(1,1)="C" THEN OPEN #1,4
   ,128,"C:":GOTO 21050
TJ 21040 OPEN #1,4,0,FILE$
YB 21050 POKE 850,7:REM Command.
RU 21060 XX=ADR(N$):YY=INT(XX/256):ZZ=INT
   (XX-YY*256)
CF 21070 POKE 852,ZZ:POKE 853,YY
OG 21080 POKE 857,255:POKE 856,12
GJ 21090 XX=USR(ADR(CIO$),16)
TU 21100 XX=PEEK(851):IF XX<>136 THEN PRI
   NT "Error ";XX;CHR$(253):STOP
IF 21110 XX=PEEK(856)+256*PEEK(857)
LL 21120 N$(XX+1)=""
VA 21130 TOTAL=XX/200
QA 21200 CLOSE #1
FL 21210 TOTAL=INT(XX/200)
EQ 21290 RETURN
EJ 21298 REM
```

Continued on page 21

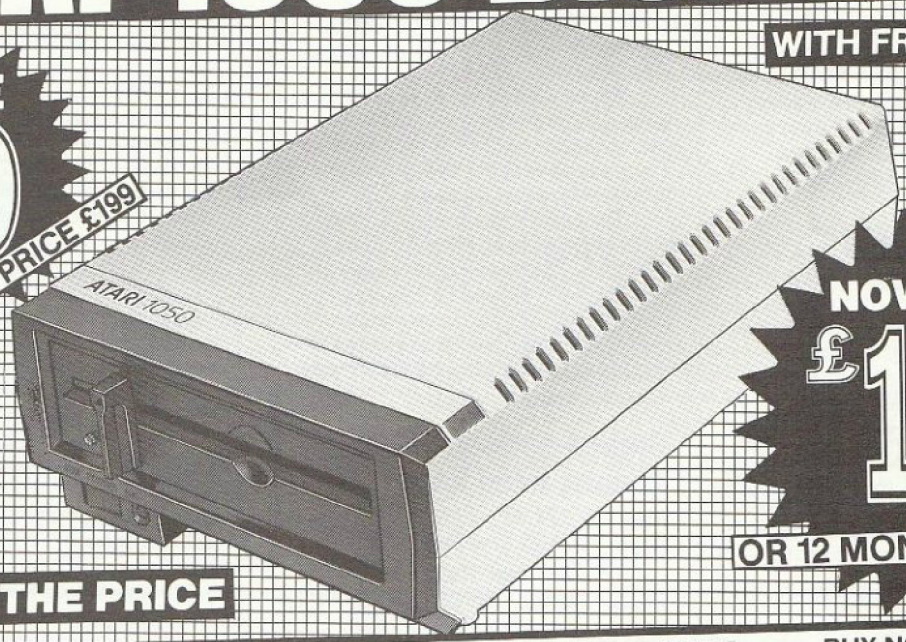
ATARI 1050 DISK DRIVE

WITH FREE SOFTWARE

WORTH £34.98

SAVE £70

PREVIOUS PRICE £199



NOW ONLY £129 inc VAT

OR 12 MONTHS AT £12.46

POWER WITHOUT THE PRICE

NEWS FOR ATARI 8-BIT COMPUTER OWNERS

Silica Shop, the UK's leading Atari specialists, based in Sidcup Kent have announced a massive drop in the price of the Atari 1050 Disk Drive. The 1050 is compatible for use with the Atari 400/800 and XL/XE computers and allows access to a range of over 450 disk-based software titles. The 1050 comes with DOS 2.5, and can also be used with other Atari Disk Operating Systems, making it compatible with the complete range of Atari Corp and Third Party software for Atari 8-bit computers. The addition of a disk drive is a great enhancement to any computer system, increasing storage facilities and cutting information access time to seconds instead of the minutes taken by the 410 or 1010 data recorders. Many professional/business programs are only available on disk and not cassette. Until now, only a small proportion of Atari Computer owners have been able to benefit from the power and speed of the 1050. Now Silica are pleased to be able to offer the 1050 at a new low price of only £129 including VAT and FREE delivery. We also offer credit facilities allowing Atari owners to purchase it over 12 months for only £12.46 per month. The total purchase price over 12 months, with interest at a flat rate of only 16%, is only 12×£12.46=£149.52 (APR 32.3%). There has never been a better time for Atari owners to buy a Disk Drive!

SUPERB TECHNICAL SPECIFICATIONS

The 1050 is a dual density disk drive. This refers to the amount of information that can be stored on a single side of a disk. Three things determine the density of a particular disk format: The number of bytes in each sector, the number of sectors per track and the number of tracks per disk. Single Density formats give eighteen 128 byte sectors on each of their 40 tracks, thus giving a total capacity of 18×128×40 or 92,160 bytes. Double Density formats give sectors that are twice as large as the single density sectors and are capable of holding 256 bytes on each sector. This yields a capacity of 184 kilobytes per disk. There is also a third format, Dual Density which is the one used by Atari's new 1050 Disk Drive with DOS 3 and DOS 2.5. It has 26 of the 128 byte sectors on each of its 40 tracks, giving a total capacity of 133 kilobytes, which after formatting comes down to 127K. The 1050 is capable of running disks formatted in either single or dual density and can run either DOS 2, DOS 2.5, or DOS 3, this makes the machine completely compatible with all of the previous old style software. The 1050 is compatible with the 400/800 and the new XL/XE series of Atari computers. However, if you run a 400 or 800 you would need to already have at least 32K RAM of memory in your computer. The 1050 has a built-in 6507 micro processor and an onboard ROM operating system for automatic standby capability. It has automatic switching from single to enhanced density modes. It has a 'Disk Busy' indicator and up to 4 disk drives can be controlled at once. Included with the Disk Drive is the Owner's Manual, the DOS 2.5 Master Diskette, Reference Manual, I/O Data cable and Mains Adaptor. The 1050 disk drive comes with everything you need to plug it in and use it immediately.

FREE SOFTWARE

The new price for the 1050 disk Drive is not the only good news for Atari Owners. The Disk Drive now comes with three FREE software titles, in addition to the DOS 2.5 Disk and Manual. The first of these is The Payoff on disk, a new adventure game in which you play the leading role. On the reverse side of this disk is a demonstration program showing Atari's amazing sound and graphics. Also included is Home Filing Manager which will help you organise your files. It allows you to catalogue and file details of books, birthdays, your stamp collection or anything else which would normally require you to use filing cards. The software which comes free with the Disk Drive carries a normal retail price of £34.98 and is as follows:

HOME FILING MANAGER	£24.99
THE PAY OFF ADVENTURE	£9.99
DOS 2.5 DISK & MANUAL	FREE
SOUND & GRAPHICS DEMO	FREE
Normal cost of FREE software	£34.98

FREE POST & PACKING

Mail Orders to the UK and BFPO are sent from Silica with post and packing FREE. We provide a 24 hour service and aim to despatch orders on the same day that we receive them.

12 MONTH GUARANTEE

The 1050 Disk Drive comes with a full 1 year guarantee. Silica will replace any faulty 1050 Disk Drive with a new unit within this period.

BUY NOW - PAY LATER

To help you to take advantage of the new low price of the 1050 Disk Drive, we have now negotiated favourable credit terms with HFC Trust. You do not need a bank account to take out credit with HFC, and no deposit is required. Repayments can be made by post, standing order, or personally at your local HFC office. If you wish to use these facilities, simply complete and return the coupon below. We will then send you further details and an application form by return of post.

OUR SPECIALIST SERVICE

We are now firmly established as the UK's NUMBER ONE Atari specialists, so when you buy an Atari computer product at Silica you will be fully supported. Our mailings keep you up to date with news of software releases and developments. Our technical support team and sales staff are at the end of the telephone line, to deal with your problems and supply your every need. We aim to stock all available Atari hardware, software, peripherals and accessories. We also stock a wide range of Atari dedicated books and magazines. Here are some of the many facilities we can offer:

- ★ FREE POST & PACKING ON MAIL ORDER
- ★ FREE NEXT DAY SECURICOR DELIVERY
- ★ INFORMATION MAILING SERVICE
- ★ TECHNICAL SUPPORT TEAM
- ★ HIGHLY COMPETITIVE PRICES
- ★ AFTER SALES SUPPORT SERVICE
- ★ REPAIR SERVICE ON ATARI PRODUCTS

ORDER NOW - OR SEND FOR A CREDIT APPLICATION FORM

Silica Shop Limited, 1-4 The Mews, Hatherley Road, Sidcup, Kent, DA14 4DX Telephone: 01 309 1111

HOTLINE!

If you can't wait to get your hands on a brand new 1050 Disk Drive for your Atari 8-bit computer and you have a credit card or charge card, phone our hotline now! We accept Access, American Express, Barclaycard, Diners Club, Mastercharge and Visa. Telephone our hotline with your card number and your address details and we will dispatch a disk drive to you TODAY! You can also call the hotline to request a credit application form if you wish to take out credit facilities with HFC Trust.

☎ 01-309 1111

1050 ORDER FORM

To: Silica Shop Ltd, Dept ATCOC 1285, 1-4 The Mews, Hatherley Road, Sidcup, Kent, DA14 4DX

Please send me ..... 1050 Disk Drive(s) at £129 each

Mr/Mrs/Ms: ..... *Initials:* ..... *Surname:* .....

Address: .....

.....

.....

..... *Postcode:* .....

☐ **I enclose a cheque:** for £ ..... (made payable to Silica Shop Ltd)

☐ **Please debit my credit card:**

CREDIT CARD NUMBER: ..... £.....

We accept: Access, American Express, Barclaycard, Diners Club, Mastercharge, Visa

☐ **I wish to pay by credit:** (12 monthly credit instalments)

Please send me written details and an application form

Kennedy Approach

48K Disk £34.95

Reveiwed by Matthew Tydeman.

Microprose Software have always been strong in the Simulation field. F15 Strike Eagle in particular making a smash into the world of Flying Simulations. Mig Alley Ace, Solo Flight and Spitfire Ace being worthy of merit in their range of cockpit simuls.

F15 and its like are a great leap forward from such so called gems as 'Floyd of the Jungle', which left quite a lot to be desired. When I heard Microprose had taken another step forward into computer-ised speach with Kennedy Approach, I knew I had to get a copy straight away.

On booting up the disk I was confronted with a title screen giving credits to the authors and then a large graphical representation of the game itself appeared. In no time at all the selection menu was up and I was ready to begin my days shift at one of the many preselectable airports of America.

There are 5 levels of skill coupled with a replay mode in which I could load a previous game. Very useful when having to stop because my latest copy of Monitor magazine has just dropped onto the doormat. A demo mode is also available giving a somewhat advanced view of upper levels and other airports. Once I had selected my level of play, I then choose my airport location which ranges from Atlanta to Fort Worth.

In order to gain access to the Control Tower you must enter a 3 figure number, get it wrong and access is denied and demo mode is automatically run. On starting my day in control, the whole airstrip is laid out before me. The current in-coming and out-going flights are displayed as little blocks with a set of small dots which represents each aircrafts height. For example, 5 dots equals 5000 feet. Aircraft flight path can be altered by joystick operation, where an arrow head is placed on the plane symbol by pressing the joystick trigger. All of a sudden, a voice comes over the intercom. "Air France EC15 turn left to 270. Descend to 5000 feet", "O.K. Roger" comes the reply. Wow, this speach is excellent and really adds realism, even atmospheric crackling of the airwaves have been included.

Its my job to bring home safely all the planes in the sky, the higher the level of play the more planes I have to control. Add Storm Clouds, Mid-air Crashes, Wrong Exits, Conflicts and Mountain ranges to contend with and this simul becomes very difficult indeed.

After a while, even on the easier levels, planes start to queue up to land and controlling them all becomes quite a task. There are no cockpit views on this one, but its still enjoyable all the same.

The price, £34.95, is a little steep, I feel, but I have been informed that US GOLD may be releasing this one as an import label at around £17.95. I cannot confirm this, but it has been mentioned to me. Besides the price, its an ideal program and a nice change from the usual shoot'em ups in my collection.

Great American Cross Country Road Race

48k Disk £14.95

Review by Matthew Tydeman

Liked-Pitstop? Thrilled with Pole Position? Great American Road Race is all anyone could want from a racing simulation plus a little bit more!

I remember the early days of Activision when I would'nt have given 10 pence for one of their programs. Over the past year now Activision, like many companies I feel, have not only pulled their socks up, they've replaced them. I'm sure you all have seen Ghostbusters (Who has'nt I hear you cry). This was the turning point for Activision even though they already had programs on the market, Ghostbusters brought all of them into the limelight and Activision now have a good name setup for them.

Road Race is similar to Pole Position but better. Boot the disk and an amazing 3D screen display rushes towards you. Up comes the credits and away you go. The idea of the game is to race across America or all the way around it in as little time as possible. You can select the starting city, and also the destination city.

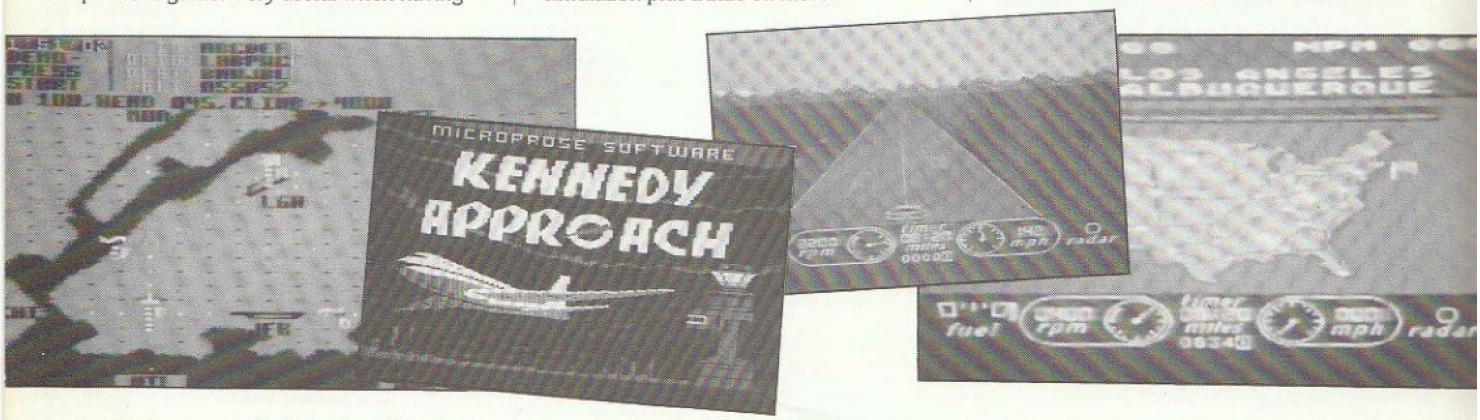
You can load different fields from disk (No, not the type with grass in!) and can play against some real rat racers who have somehow managed to score impossible completion times. Eight fields are allowed, each one a little harder than the last. Select your field and destination from the map of America and its states and plan your route. Watch out for the Storm clouds, Fog patches and Road Works – these will slow you down. Avoid the road works by taking a different route but be carefull – this new route may be a chilly one and the roads may be covered in ice. Route selected, you begin the game. Change the gears using the Joystick and Trigger to depress the clutch and zoom your car off. Keep an eye on the rev counter, if the revs are too high the engine will blow and you'll have to pump the car to the nearest fuel stop for repairs. The fuel stops come past fast so keep at a steady speed once you receive the signal. Watch out for your fuel guage too, this has the tendency to rush down to zero rather fast so you'll have to keep a constant eye on this one. The road is bendy – and depending on the route chosen it may be long. On the horizon are detailed little scenery drawings which change depending on the route being driven.

Avoiding the cars, which incidently are'nt that good and resemble a squashed Skoda Estelle to me, was hard at first. After a few games these multicoloured death traps were gone and passed – thanks to my superduper accelerator which enabled me to shoot through the cars at a nice, steady true to life (!) speed of 240mph. At these speeds that blasted policeman in his suped up Mustang did'nt stand a chance – I was gone and Burnin' Rubber as those American friends of ours say. If I were apprehended at a catchable speed however he did'nt like it and time was taken to give me a ticket.

After racing the chosen route the horizon disappears after the message 'City Limits'. The traffic gets noticeably heavier and my speed noticeably slower! Up comes the town with its key representation of the on coming city (i.e New York is depicted by the silhouette of the Statue of Liberty). You are welcomed into the city and the race continues.

If, or when you complete the whole race you get the chance to join those crazy rat racers up there in the hidden tracks of your floppy disk. Your name then becomes a part of the unique fame of the Great American Cross Country Road Race (Say that when you're drunk!).

This is a good program. If you like racing simulations this is undoubtedly a must for everyone and at £14.95 its a scratch off the wallet. Activision! Keep 'em coming.



USER GROUP SOFTWARE

Software Librarian - Roy Smith

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

3 FOR 1 EXCHANGE

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

DONATION SCHEME

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

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

THE LIBRARY SOFTWARE SERVICE IS FOR MEMBERS ONLY

LIBRARY SOFTWARE TITLES

Listed below are the software titles received by members for inclusion in the library since issue nine was published. As the library now contains over 300 programs it is getting a bit too large to keep on printing the entire list. Eventually it would probably take over the whole magazine and there would be no room left for the articles and program listings. For those of you who are new members and do not know what is available from the library then send for a photocopy of the complete list which is available from the librarian. There is a small charge for this service to cover photocopying costs. If you would like a list please send 50p and a SAE for return.

Games

ANIMAL
by M.Maestranzi – Finchley.
Guessing game where you supply the clues and the computer gives the answer.
Runs in 16K Cassette or 32K Disk min.

BIRD SHOOT
by Bob Askew – Northampton.
As the Farmer you must prevent the birds from landing in your field.
Runs in 16K Cassette or 32K Disk min. Not XL/XE compatible.

CIRCUITS
by Stephen Taylor – London.
Pacman style game with excellent graphics.
Runs in 32K Cassette or Disk min. XL/XE machines only.

MERCHANT SPACEMAN
by Tony Search – Southend.
Intergalactic trading simulation.
Runs in 48K min. Dual Density Disk only.

PCB PARANOIA
by Steve Hillen – Southend.
Can you outlast the maniac track makers?
Runs in 16K Cassette or Disk min.

Adventure Games

SPECTRE OF CASTLE DOOMROCK
by Scott Smith – Inverness.
Make your way through the castle rooms looking for gold.
Runs in 32K Cassette or Disk min.

DIAMOND 1 & 2
by Scott Smith – Inverness.
Two adventures in which you must find the Phia Diamond and destroy a Time Machine.
Runs in 16K Cassette or 32K Disk min.

Home Entertainment

DAY FINDER
by Paul Rixon – Shefford.
Find out what day of the week you were born on with this program (if born before 1753).
Runs in 16K Cassette or Disk min.

GOLF
by Scott Smith – Inverness.
Sharpen up your golf technique.
Runs in 16K Cassette or 32K Disk min.

MATCH
by M.Maestranzi – Finchley.
Match the shapes to score.
Runs in 16K Cassette or Disk min.

TOUCH TABLET SHOW
by Mark & Brian Christian – Wirral.
11 pictures created using Atari Touch Tablet. Loaded from DOS and includes Fader program. Touch Tablet not required.
Runs in 48K min. Disk only.

Utilities

ALPHACOM 81 SCREENDUMP
by M.Maestranzi – Finchley.
A Touch Tablet or Mode 15 screen dump utility for the Alphacom 81 printer.
Runs in 16K Cassette or Disk min.

AUTORUN
by Paul Rixon – Shefford.
Cassette utility provides title screen whilst program loads.
Runs in 16K Cassette min.

BEELINE
by Terry Whitney – Bromley.
BASIC program to create an AUTORUN.SYS file that will run a BASIC program of your choice on boot-up.
Runs in 16K min. Disk only.

CANON FONT
by J.T.Boyle.
Define a font and download it to a Canon PW-1080A printer.
Runs in 48K min. Disk only.

DELETE/AUTOLINE
by K.Vaughan – Basingstoke.
Useful utility giving automatic line numbers and delete line facility.
Runs in 16K Cassette or Disk min.

★★★ **STAR PROGRAM** ★★★

DISK JACKET
by Alan Goldsbro – Leeds.
This program prints out a shape on your Epson printer that can be cut out and made into a disk jacket.
Runs in 16K min. Disk only.

★★★★★★★★★★★★★★★

EASY SELECT DISK DIRECTORY
by Stephen Taylor – London.
Gives quick and easy access to disk files.
Runs in 16K min. Disk only.

SKETCH ATARI
by Lee Brown – Southend.
Graphics 8 sketch pad.
Runs in 32K min. Disk only.

SNOOKER LEAGUE
by Dick Fake.
Keep records of team scores and division positions.
Runs in 48K min. Disk only.

Education

INVENTIONS
by Paul Rixon – Shefford.
Quizgame asking who invented what?
Runs in 16K Cassette or Disk min.

Music

MUSIC 5
by Jeff Davies – Llandeilo.
3 tunes for use with Atari Music Composer. Star Wars, Hatari and Jingle Bells.
Runs in 16K Cassette or Disk min.

TOP TEN

1	(–)	**Chase**	**Grahame Fairall**
2	(–)	**Composed Writer**	**Larry Farmer**
3	(1)	**Usercomp**	**Trevor Skeggs**
4	(–)	**Johnnys Paintbox**	**Stan Ockers**
5	(–)	**Superfruit 2**	**Grahame Fairall**
6	(–)	**Masters DOS**	**Matthew Tydeman**
7	(–)	**Assembler**	**Chris Rutter**
8	(–)	**Pharoahs Tomb**	**Sydney Brown**
9	(5)	**Stoneville Manor**	**Nigel Haslock**
10	(–)	**Idol Island**	**J.P.Crackett**

Adventure into the ATARI

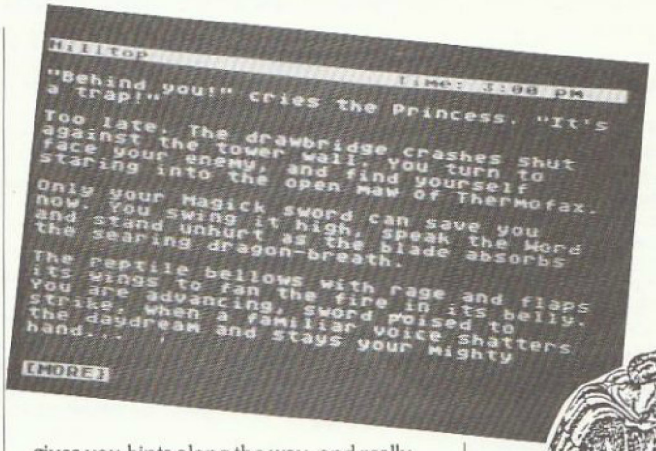
by Steve Hillen

Wishbringer

48k Disk £29.95

Wishbringer is, I believe, the sixteenth title in Infocom's series and rated at introductory level. It has been written by Brian Moriarty, the ex-technical editor of 'Analog'. The game shows a lot of influence from his 'Crash-Dive' game published in 'Analog', some of it good and some of it not so good. Although there may be cosmetic quibbles, the actual adventure itself is imaginative and exciting. You start the game as a lowly post-man in the quiet village of Festeron. Your first job is to deliver a mysterious envelope to the old lady at the Magick Shoppe. Once there, you discover that her cat, Chaos, has been kidnapped by the 'Evil One' who has designs on a stone called Wishbringer. This stone will grant certain wishes for its owner, as long as certain objects are present as a focus. For example, to wish for darkness, you must have recently swallowed some grue's milk. As you leave the shoppe, the landscape has changed into a magical realm where trolls guard bridges and an evil-looking vulture follows the wicked Boot-patrol who enforce the curfew. If you are unlucky enough to get caught by them, you will be slung in jail, eventually to be tortured to death by none other than the Evil One herself. However, if you manage to find and return the cat, Chaos, then the stone will be yours and you can set about your task of defeating the Evil One.

This is definately an easier adventure than most of Infocom's games. The game



gives you hints along the way, and really spells it out to you if you do something right. Nevertheless, it is still challenging and has the option to be played with or without Magick. The prose is up to their usual standard, and it really seems more like a novel than a game. Also, the adventure benefits from having a totally original theme, as well as the standard Infocom humour, an example of which is the scene and mail-box from Zork. Now the grudges. I don't like the keyboard handler Brian has brought in from Crash-Dive. What is the point in being able to type in long sentences only to find that you cannot use the cursor keys to correct a spelling mistake at the beginning? It's as bad as using the BBC's line editor! Also there's no buffer, so you have to wait while the program accesses the disk. Despite this unwelcome change to Infocom's usual style, the actual game is still excellent.

Finally, one to watch out for if you're lucky enough to own a 520ST is Infocom's 'A Mind Forever Voyaging'. As the advance blurb tells you, you are a computer brought up as a human child. Your job is to enter a world of the future and examine the effects of various world-wide policies designed to save society from chaos (not the cat!). Infocom claim that the game understands over 1700 words and 300 types of sentences! We'll just have to wait and see.

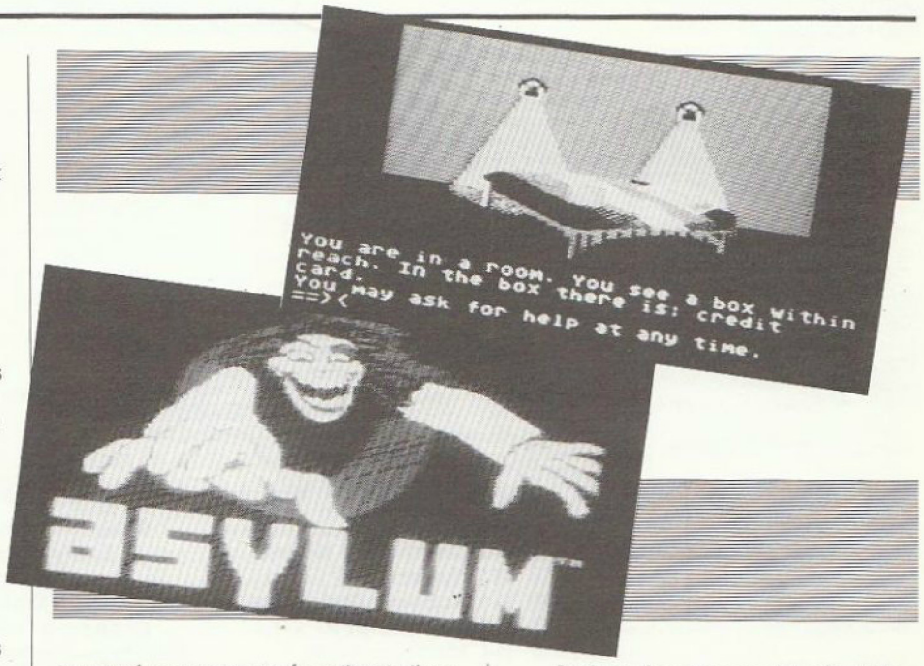
Asylum

48k Disk

Well, the Atari version sure took a long time to come out, but was it worth the wait? It's a pretty old adventure by now but on the whole it is one of the better ones. The graphics are excellent, it is after all a very graphics-orientated game. Also the parser will understand a variety of sentences, and the theme of the game is quite amusing too.

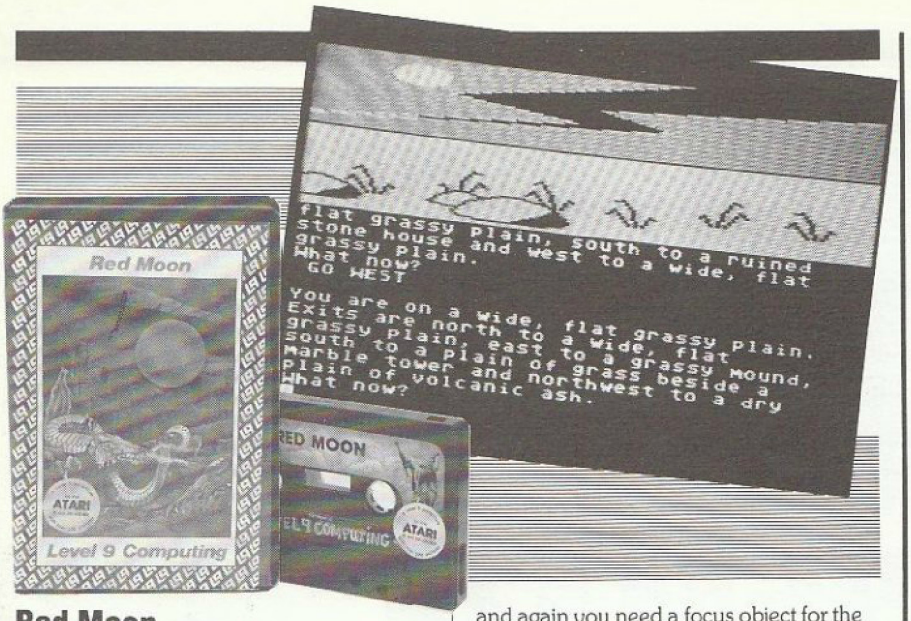
You play the part of an unfortunate 21st century computer adventurer who has lost sight of what is fact and what is fiction. Subsequently, you were taken away to the Asylum to recover. The only way you can prove that you are fit to rejoin society is by escaping from the Asylum. As one of the mad inmates suggests, the only way to do that is to disguise yourself as a doctor by wearing a white coat, and leave by the doctors' exit. The problem is, where is the white coat and the exit?

You start in a small room which opens onto a fiendish maze of lookalike corridors. Unusually, the movement is controlled by cursor keys, and the 3-D animation, although not of 'Wayout' quality is still pretty impressive. Doors open and mad



inmates leap out on you from time to time to keep the action flowing. Another unusual feature is the provision of a list of words that the game understands, not that you will need to use many.

In short, the game consists of mainly wandering the corridors as in the classic maze games and, to a logical adventurer used to more standard games, may well be disappointing.



Red Moon

32k Cassette £6.95

Level 9 have been busy recently, for this latest release includes graphics. It's no mean feat to have pictures and descriptions for over 200 rooms in a 32K cassette game! The game does take rather a long time to load, however. At first I was surprised that graphics had been included for they can slow the game down. Level 9 have programmed the game superbly though, for the picture-drawing and text-input are completely independent. This means that you carry on typing commands and watching room descriptions while the scene is being drawn, unlike 'Stranded'. I'm extremely impressed.

The actual adventure is superb too. As the story goes, long ago the moon glowed red and provided power for Magik to be cast. As time passed, the moon glowed less and less brilliantly, until in desperation, the last Magicians decided to build a new moon. They made the Red Moon Crystal, and placed it in a tower to preserve Magik. Unfortunately, it was stolen, and you're the one who must recover it.

As in Wishbringer, you can cast spells and again you need a focus object for the magic to work. That aside, the two games are vastly different. I was lost for hours in the hundreds of rooms and vaults below Castle Cakabol, where supposedly the crystal may be found. There are a total of 9 treasures waiting for you, although the number of nasty Mediaeval creatures waiting to pounce on you makes progress difficult. You will find yourself doing combat with the likes of giant rats, cloaked statues and barbarous blacksmiths. It helps to have a weapon of some sort and armour, although the best solution is to attack them magikally if you have the dagger as a focus. These frequent battles keep the game very lively.

Summing up, Level 9 has taken a giant leap forward with this game. The whole program is superb. Incidentally, the packaging is also improving, would you believe they enclose a red balloon? Look out for new releases 'The Worm in Paradise', the sequel to Snowball and Return to Eden, and 'The Diary of Adrian Mole' (I suppose it had to become a game sometime!), and if you don't already have a collection of Level 9 games, go out and buy them! You won't be sorry.

Future Paths

Unfortunately, due to a change in circumstances, I no longer have much time to sit down and play adventures, so apart from the odd one or two reviews per issue, the Adventure Column is going to change, starting from the next issue. The Q and A section will be continued, but you may have to wait a little longer for replies. Of course, any adventure reviews you care to submit can still be published within the column.

Instead of reviews, I'm considering starting a series on writing your own adventures, as I've already had quite a few letters asking for advice and tips on adventure writing. At the moment, I have a rough idea of what sort of adventure to write; one which I haven't seen anyone else do yet. Nevertheless, it's always nicer if you write in telling me exactly what sort of game you'd like to see produced. Now's your chance to affect what appears in the column for the next 3 or 4 issues!

The questions I'm going to ask you are :–

1. What sort of parser are we to have? A simple Scott Adams style VERB-NOUN one or a more complicated LEVEL 9 style one capable of analysing simple sentences ?
2. What sort of screen display and room descriptions are we to have? A split-screen multi-windowed one such as WAXWORKS, or a normal screen with a status line such as ZORK. The window approach does limit the length of the room descriptions. Obviously the 16K crowd will opt for the simpler choices in order to fit a game into their system.

Anyway, please let me know your thoughts on this idea of writing our own adventure.

There are still quite a few unanswered questions. If you can solve any of them, I'd be pleased to hear from you. There are also quite a lot that have been answered. If you need to know the answer to one of the questions below, type in Listing 1 using BASIC, and add the line 20 found below the relevant question. Once you RUN the program, the solution to the problem will be deciphered.

Listing 1

```
10 DIM A$(100)
30 FOR A=1 TO LEN(A$):A$(A,A)=
CHR$(ASC(A$(A,A))-4):NEXT A:?A$
```

Answered Questions

Zork 2

How do you move the menhir?

```
20 A$="YWI$XLI$[M^EVHW$[ERH"
```

By Gary Cheung

What is the rose for?

```
20 A$="MXW$NYWX$E$KMJX"
```

By Gary Cheung

How do you use the balloon?

```
20 A$="YWI$LSX$EMV"
```

By Gary Cheung

Waxworks

Where is the well?

```
20 A$="[EMX$RIEV$RS$[EMXMRK$
WMKR"
```

By D. Harding & Steven Trick

What do you do with the mummy?

```
20 A$="YWI$GLEVQ$ERH$VIERMQEX
I$MX"
```

How do you get past the trapdoor?

```
20 A$="JM\$MX$JVSQ$FIPS["
```

Feasabilty Experiment

Can you kill the dragon?

```
20 A$="]IW$YWI$M\MSR$WLMIPH$ER
H$IFSR]$WTIEV"
```

By Peter Lister

Zork 1

What is the machine for?

```
20 A$="TYX$GSEP$MR$ERH$YWI$
WGVI[HVMZIV"
```

What do you do with the egg?

```
20 A$="KMZI$XS$XLMIJ$XS$QIRH"
```

Pirate Adventure

How do you get past the snake?

```
20 A$="YWI$GVEGOIVW$ERH$TEV
VSX"
```

The Count

How do you break the lock on the coffin?

```
20 A$="[EMX$XMP$HEVO$ERH$YWI$
JMPI$JVSQ$SZIR"
```

Dungeon Adventure

How do you enter the troll room?

```
20 A$="IEX$MRZMWMFPI$QYWLVS
SQW$RIEV$XLEX$VSSQ"
```

By Peter Lister

Unanswered Questions

Savage Island 2

Can't kill dinosaurs nor change back from Neanderthal.

Escape From Pulsar 7

Can't remove cups from captain's cabin ceiling.

Curse of Crowley Manor

Have combination but can't get past monster in numerical lock room.

Demon Knight

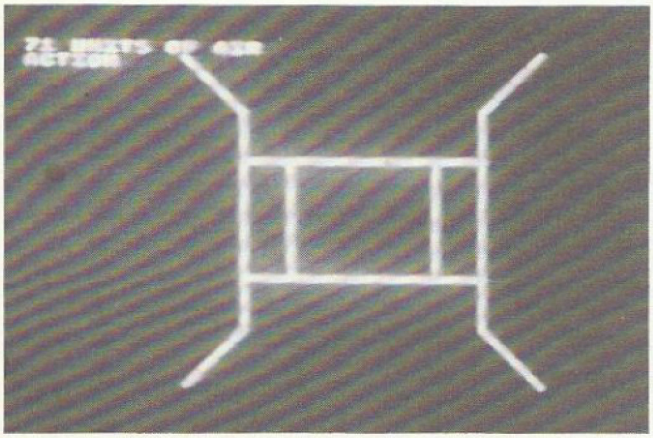
Can't get past jewel-encrusted room.

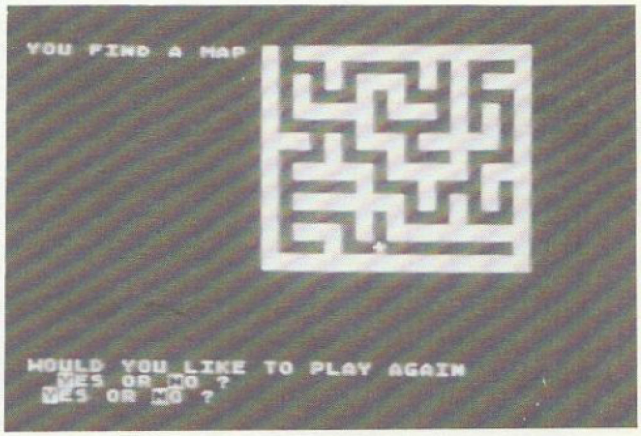
Stranded

Can't progress from desert to forest.

Treasure of the Golden Reef

How do you get a response from the cow?
How do you get over the wall?





3D-MAZE

by Tommy Taylor-Cardiff

This is a short BASIC program which puts you into a 3 dimensional maze. Your task is to escape before your air supply is exhausted. Use the joystick in port 1 to play the game. By moving the joystick backwards, you'll be facing the way that you came.

Type in the listing below. If you're not using KEYO then just ignore the two characters before each line number.

The actual program may look familiar to some of you, as it was converted from a PET program published many moons ago.

(Runs in 16K cassette, 24K disk)

```
IE 1 DIM W(10):DIM M(10):DIM A$(1):GOSUB
   2000
LG 3 GOSUB 5000
LZ 4 GOSUB 7000
QM 5 GOTO 900
SA 10 Y=PEEK(TC-U):Z=PEEK(TC-U+LL):IF ABS
   (U)=1 OR DC=0 THEN 30
XQ 20 IF Y=0 AND Z=0 THEN FOR A=TC-U+LL T
   O BC-V-LL STEP LL:POKE A,W3:NEXT A
ES 30 POKE TC,TW:POKE BC,BW:B=B+1:IF B<DW
    THEN TC=TC+U:BC=BC+V:GOTO 30
LB 40 IF ABS(U)=1 THEN POKE TC,T1:POKE BC
   ,B1
ED 50 IF TC=BC-LL THEN RETURN
DF 60 FOR B=TC+LL TO BC-LL STEP LL:POKE B
   ,SW:NEXT B:RETURN
XW 70 F=ML:U=LL+1:V=1-LL:T1=1:B1=4:SW=66
HA 80 FOR X=4 TO 6 STEP 2:TW=71:BW=70:B=0
   :IF X=4 THEN GOTO 100
QL 90 F=MR:U=LL-1:V=-LL-1:SW=86:T1=2:B1=3
   :TW=70:BW=71
ZJ 100 LM=M(F):LD=W(F)
YE 101 IF F=3 AND OP=EC THEN GOTO 140
EG 105 IF ZZ-LD*INT(ZZ/LD)=0 THEN TC=M(X)
   :BC=M(X+1):GOSUB 10:GOTO 130
GE 110 TW=77:BW=78:U=U-LL:V=V+LL:N=PEEK(O
   P+LM)
JD 120 IF N-D*INT(N/D)=0 THEN TC=M(X)+DW*
   LL:BC=M(X+1)+DW*(-LL):GOSUB 10
DN 130 IF ABS(U)=1 THEN U=U+LL:V=V-LL
MW 140 M(X)=M(X)+DW*U:M(X+1)=M(X+1)+DW*V:
   NEXT X
AS 150 IF M=3 AND OP=EC THEN RETURN
JH 160 IF ZZ-D*INT(ZZ/D)=0 THEN GOTO 180
OR 170 DW=DW-1:OP=OP+M(M):ZZ=PEEK(OP):DC=
   DC+1:IF DC<CD THEN GOTO 70
SZ 175 IF DC=CD THEN RETURN
QG 180 R=PEEK(M(4)-1)
PT 185 IF R=0 THEN M(4)=M(4)-1:M(5)=M(5)-
   1:IF M(4)<>TL+DW*LL THEN GOTO 180
QE 190 R=PEEK(M(6)+1)
LJ 195 IF R=0 THEN M(6)=M(6)+1:M(7)=M(7)+
   1:IF M(6)<>TR+DW*LL THEN GOTO 190
PJ 200 FOR TC=M(4) TO M(6):POKE TC,W1:NEX
   T TC
TB 210 FOR BC=M(5) TO M(7):POKE BC,W2:NEX
   T BC:IF K=1 THEN GOSUB 598
ZD 211 RETURN
PY 240 ? OX;" UNITS OF AIR"
EF 242 DW=4
VP 243 IF SS=1 THEN GOSUB 7030
LU 245 ? "ACTION"
GP 246 AZ=STICK(0):IF AZ=15 THEN GOTO 246
MF 247 IF AZ=10 OR AZ=6 OR AZ=5 OR AZ=9 T
   HEN 246
GA 250 NP=P:ZZ=PEEK(P):IF AZ=14 THEN NP=P
   +M(M)
OL 260 IF NP=P THEN GOTO 320
EK 270 IF NP=EC THEN K=1
NZ 285 IF ZZ-D*INT(ZZ/D)<>0 THEN P=NP:? "
   YOU MOVE":GOTO 380
FG 300 IF RND(1)>0.9 THEN ? "YOUR SUIT RI
   PS WHEN";:OL=OL+1
MP 310 ? " YOU HIT A WALL":GOSUB 3000:GOT
   O 410
RS 320 NM=M:IF AZ=7 THEN NM=M+1
FH 330 IF AZ=13 THEN NM=M+2
FJ 340 IF AZ=11 THEN NM=M+3
NB 350 IF NM=M THEN GOTO 240
NQ 360 IF NM>3 THEN NM=NM-4*INT(NM/4)
IS 370 M=NM:? "YOU TURN"
SY 380 D=W(M):ZZ=PEEK(P):OP=P:ML=M-1:MR=M
   +1
EG 390 IF ML<0 THEN ML=3
JR 400 IF MR>3 THEN MR=0
QA 410 OX=OX-OL:IF OX<0 THEN ? "YOU DIE."
   :GOSUB 4000:K=1:GOTO 600
VJ 420 T=T+1
LE 430 DC=0:GOSUB 2000:GOSUB 70
LN 435 IF P=MC AND RM=0 THEN RM=1:GOSUB 6
   00
DZ 440 M(4)=TL:M(5)=BL:M(6)=TR:M(7)=BR:GO
   TO 240
SZ 500 P=P+1:IF P>EM THEN P=SM
IQ 510 A=INT(RND(1)*4):DC=0
XZ 520 A=A+1:DC=DC+1:IF DC>3 THEN 500
KH 530 IF A>3 THEN A=0
JH 540 M=P+M(A):IF M<SM OR M>EM THEN 520
LY 550 CP=PEEK(P):CM=PEEK(M):IF C>0 AND C
   P=210 THEN P=M:GOTO 510
EP 560 TM=M-SM:IF (CP=CM OR CM<210) AND C
   >0 THEN 520
AI 570 ME=TM-L*INT(TM/L):IF (ME=0 AND M(A
   )=1) OR (ME=6 AND M(A)=-1) THEN 520
BL 580 OD=INT(15/W(A)):CP=CP/W(A):POKE P,
   CP:CM=CM/OD:POKE M,CM
RX 590 P=M:C=C+1:IF C<H THEN 510
AI 595 RETURN
WX 598 A=SC+98:GOSUB 2000:? "YOU ESCAPE "
   :GOTO 605
```

3D MAZE

```
VE 599 STOP
DM 600 A=SC+98:GOSUB 2000:? "YOU FIND A M
   AP"
QN 605 MS=A:DC=SM:DD=SM+G:GOSUB 7060
SW 610 FOR B=DC TO DD:FOR C=0 TO 3:CB=PEE
   K(B):CB=CB-W(C)*INT(CB/W(C))
BU 620 AA=1:IF C=1 OR C=3 THEN AA=LL
IO 630 IF C>1 THEN AA=-AA
JZ 640 BB=LL/AA:P2=A+AA:P1=P2+BB:P3=P2-BB
DG 645 IF B=P THEN POKE A,W4
MW 650 IF CB=0 THEN POKE P1,W3:POKE P2,W3
   :POKE P3,W3
DI 660 NEXT C:A=A+2:NEXT B:DC=DC+L:DD=DD+
   L:A=MS+(2*LL):MS=A
BN 670 IF DD<=EM THEN 610
UO 680 GOSUB 4000:IF K=1 THEN GOSUB 7060:
   GOTO 6000
GN 690 A=TL+210:RETURN
DI 900 GOSUB 2000:? "YOU HIDE IN AN ALIEN
    MAZE TO AVOID"
QX 905 ? "CAPTURE BY YOUR DEADLY ENEMIES"
YT 910 ? "FIND THE EXIT BEFORE YOUR AIR R
   UNS OUT"
QY 950 TL=SC+92:BL=SC+932:TR=SC+114:BR=SC
   +954:LL=40:DW=4:CD=4
GC 955 W1=77:W2=78:W3=6:W4=5
JY 960 W(0)=5:W(1)=7:W(2)=3:W(3)=2:L=INT(
   RND(1)*5+6)
UG 970 M(0)=1:M(1)=L:M(2)=-1:M(3)=-L:M(4)
   =TL:M(5)=BL:M(6)=TR:M(7)=BR
SU 980 W=INT(RND(1)*5+6):H=L*W-1:G=L-1:C=
   0:DC=0:T=0:RM=0
EY 995 SM=(PEEK(756)-4)*256:EM=SM+H:FOR A
   =SM TO EM:POKE A,210:NEXT A:P=SM+INT(R
   ND(1)*H/2)
RC 1000 EC=SM+INT(RND(1)*L):MC=INT(RND(1)
   *H)+SM
PV 1010 OX=INT(H/3)+1:IF P<OX THEN OX=-OX
EO 1015 OX=OX+H:OL=1:GOSUB 500:GOSUB 2000
WT 1030 M=0:NP=P:CE=PEEK(EC):CE=CE/W(3):P
   OKE EC,CE:GOTO 380
GQ 2000 ? CHR$(125):RETURN
WR 3000 FOR X=0 TO 80:NEXT X:RETURN
PR 4000 FOR X=0 TO 200:NEXT X:RETURN
HT 4999 END
WS 5000 GRAPHICS 0:POKE 710,0:POKE 752,1:
   SC=PEEK(88)+256*PEEK(89)
MN 5010 FOR A=1536 TO 1569:READ D:POKE A,
   D:NEXT A
BK 5020 DATA 104,104,133,204,104,133,203,
   169,0,133,205,169,224,133,206,162,4,16
   0,0
SR 5030 DATA 177,205,145,203,136,208,249,
   230,204,230,206,202,208,242,96
UL 5040 X=USR(1536,256*(PEEK(89)-4)):POKE
    756,PEEK(89)-4
JH 5050 FOR A=8 TO 55:READ D:POKE PEEK(75
   6)*256+A,D:NEXT A
RL 5060 DATA 255,255,3,3,3,3,3,3,255,255,
   192,192,192,192,192,192
PW 5070 DATA 192,192,192,192,192,192,255,
   255,3,3,3,3,3,3,255,255
VQ 5080 DATA 0,24,26,126,28,28,20,54,255,
   255,255,255,255,255,255,255
BE 5090 RETURN
EN 6000 POSITION 2,22
BT 6010 ? "WOULD YOU LIKE TO PLAY AGAIN
          ";
IG 6020 ? " Yes OR No ";:INPUT A$:A$=A$(1
   ,1)
PL 6030 IF A$<>"Y" AND A$<>"N" THEN GOTO
   6020
DO 6040 IF A$="Y" THEN K=0:GOSUB 2000:GOT
   O 4
CE 6045 GRAPHICS 0
FH 6050 END
LJ 7000 SS=0:? "DO YOU WANT SOUND ( Y OR
   N) ";:INPUT A$
SH 7010 IF A$="Y" THEN SS=1
BC 7015 RETURN
NH 7030 FA=INT(OX*2/3+170)
HX 7040 SOUND 0,FA,10,3:SOUND 1,FA-3,10,3
   :SOUND 2,FA-6,10,3:RETURN
ZS 7060 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUND
    2,0,0,0:RETURN
```

Opening Out *Continued from page 13.*

```
EO 21299 REM
JT 32000 REM Fast String Search Routine
NZ 32001 REM **From Issue 7 of Monitor*
VE 32002 REM ======By Keith Mayhew=====
YW 32010 DIM S$(109):RESTORE 32050
VD 32020 FOR XX=1 TO 109:READ YY
YO 32030 S$(XX,XX)=CHR$(YY):NEXT XX
DT 32040 RETURN
VB 32050 DATA 104,104,133,204,104,133,203
   ,104,141,241,6,104,141,240,6,104,133,2
   06,104,133,205,104,104,141,242,6,240
VR 32051 DATA 74,169,1,141,243,6,169,0,14
   1,244,6,238,240,6,208,3,238,241,6,160,
   0,177,203,209,205,208,18,200,204,242
SB 32052 DATA 6,208,244,173,243,6,133,212
   ,173,244,6,133,213,96,104,238,243,6,20
   8,3,238,244,6,230,203,208,2,230,204
VK 32053 DATA 173,244,6,205,241,6,208,208
   ,173,243,6,205,240,6,208,200,169,0,133
   ,212,133,213,96
```

Using The Program

As I mentioned earlier, the major drawback in holding all our data in memory is that it restricts the number of records we can have in our file. The exact number will depend upon how much memory your computer has, (as well as the type of DOS you use if you own a disk drive) but you should find that a 48K machine will allow you to store around 130 records.

There is, however, one very significant advantage gained from having all our data in memory – it enables us to search for and locate records very rapidly. Take a look at lines 4000 to 4999, for here you will see a routine to find a particular record. The program asks you to enter part of the name and address you wish to find, the idea being that you enter something which you know is in the record, and which is reasonably unique. The program then hunts through the data string and displays any records containing the sequence of characters you entered.

To do this, the program uses a machine code program designed to perform the same function as the INSTR command available in most versions of Microsoft Basic, but sadly is missing from Atari Basic. This is used to find the occurence of one string inside another, larger string. The machine code routine I have used here was written by Keith Mayhew, and was originaly published in Issue 7 of this magazine, and it is an extreemly usefull utility.(Note: Issue 7 is completely sold out, but a photocopy of Keith's Ultra Fast String Search article is available if you send 50p and an SAE to the magazine address). This machine code program is entirely relocatable, so I have placed it into a string and called it S$.

Once again, you will notice that the USR command is used to call the routine, but this time we must pass through it the address in memory and the lengths of both the larger string and the smaller string respectively, you will see how this is done on line 4080. The routine returns the first occurence of a match from this program line, but if this is not the one we want then we might have to look for a further match. To do this we use the USR function to call the search routine, but this time we need to jump into it at a different address, to be precise 71 bytes from the start of S$. Hence the USR instruction on line 4170

You should find that in use you will be able to locate an address in a fraction of a second – quite an astonishing feat for such a small program!

In Conclusion

I hope that this program is helpfull to you in demonstrating that string manipulation is a very important part of file handling, and that you need to be fluent in the juggling of strings before you can begin to write efficient data processing programs. It is, of course, far from being the most efficient of database programs, and I would like to hear of any modifications or additions you may have made. In the next part of this article I will cover some of the more specialised commands available to disk-drive users, and give an updated version of our mini data-base. Bye for now.

WHAT'S MIDI?

by Michael Stringer Part Two

In the first part of this series I introduced you to the history, implementation and uses of the Musical Instrument Digital Interface. This is a system comprising a language and the interface. In Part 1 I gave a brief review of the structure of the interface and in Part 2 I will look, in great detail at the language of the system.

Since I wrote the first part there have been over fifty new musical instruments and ancillary equipment released on to the home market. One very interesting addition to the manufacturers list is that of Fairlight. This is regarded by most devotees as being the very best of synthesisers currently available. The fact that Fairlight have adopted MIDI as their standard is very interesting because, initially, they were going to develop their own form of MIDI. In their opinion, the MIDI Standard was too slow. This you will remember, I hope, is 31.25 Kbaud, i.e. 31,250 bits per second. Not as I read recently in another Atari orientated magazine 31.25 Megabaud (31.25 million-bits per second !!) which is over 23 times faster than the data transfer rate of the 520ST Hard Disc (coming soon, I hear). The baud rate that Fairlight were looking at was in excess of 90 Kbaud. This is the sort of speed that the 'knowledgable people' regard as a minimum speed. It is almost certainly the vast amount of ready available ancillary equipment centered around 31.25 Kbaud which persuaded them to adopt it as their standard also.

I have also taken delivery of a pre-release 520ST which is sitting in front of me at this moment. 'Sitting' is the operative word, it doesn't do very much. The machine came with GEM and some peculiar language called LOGO, this version developed by Digital Research. BASIC was supposed to have been included, but its release has been delayed and it should be sent to me as soon as it is available. I expect that at the time you read this I will have Basic in my hands, well I'm keeping my fingers crossed. There was I, hoping to dazzle you all with some examples of short MIDI demos, but I am afraid we will all have to wait.

I can assure you that it is extremely frustrating, to say the least! Never mind, I suppose I could learn all about LOGO in the meantime, except I cannot find any books on the subject in the local libraries, nor in the local College of Technology Libraries. Even Digital Research have not replied to a written request for help. Getting a pre-release date model to try and develop some decent software for MIDI enthusiasts to cut their teeth on has so far been a totally frustrating affair.

My synthesiser, by the way, is the Yamaha DX7. Appreciating this is most important, when we get to it later in this article. It is this particular instrument that I will be concentrating on for the simple reason that I am more familiar with this than any other one.

Before I take you into the structure of the MIDI language, I think this is a suitable place to give a brief description of the DX7. Most of you will have seen this instrument, and I am certain that all of you will have heard it action, although it is likely that you were unaware that the sound heard was created by it!

The DX7 is a programmable algorithm Frequency Modulation synthesiser. The sound is created from sine waves created by Operators, of which there are six. These Operators can interact with each other in set patterns. Thirty two patterns are available, and it is this pattern that is called an Algorithm. There are two types of Operator, Modifiers and Carriers. The colour of the sound is developed from the frequency ratio of the Modifiers and Carriers and then is fed, via various filters and oscillators, to the amplifier. The sound characteristics depend on the shape of the algorithm. Figure 1 shows just three types from those available.

The modulation index governs the depth of tone, which is the mellowness or brilliance. The envelope of the carriers determines the overall 'shape' of the sound. The output can be visualised as flowing down the algorithm, through the Modifiers and out through the Carriers.

To set up a sound there are over 140 controllers, most of which can be set in the range 0 to 99! Each controller is under MIDI or Manual control. The number of different sounds possible with this instrument is over 10,000,000,000,000,000,000!!. It sounds very complicated, doesn't it? It is. It took me three months to learn how to program this instrument efficiently.

The difficulty that some people have in programming the DX7 is one of the reasons that good MIDI software is in great demand. It takes me two days, or thereabouts, to create a new sound from

scratch. Once a new voice is created it has to be stored, which means transfering it to a RAM pack. These packs cost £70 each! The maximum number of voices that can be stored in each pack is only 32. Storing voices, or sounds, is very expensive. The data, via MIDI, occupies something approaching 200 bytes. Saved on a disk, the number of sounds is vast. In fact it would be financially worthwhile having the 520ST just for building a library of sounds. Each sound can be allocated a name, devised by the computer, based upon the instruments the sound resembles. The MIDI standard for voice naming are ASCII characters, of which ten can be used. The number of ASCII characters available is over 200. If a *, for instance, represented a string sound, a > represented a keyboard instrument such as a piano, a sound resembling a combination of string and piano could have as a prefix in the title * > 702. As the sound library increased, it would be a relatively simple task to include a sort within the program in order that sound types could be filed as per a database. Retrieval would then be a very simple and rapid means of filling the data bank in the synthesiser. When creating a sound, a complete sound-creation section, such as the Envelope, or Keyboard Scale could be allocated to one screen. The 520ST can have five screens visible at any one time. The visual and audible effect, including a graphical representation of the wave form, would enable sounds to be created in a very short space of time.

The instrument comes with two ROM packs, each containing 64 sounds per pack. These sounds can be rapidly edited, the easiest way being to simply change the Algorithm. We now have over 4000 different sounds (64 X 32). Many sound very similar, but of these 4000 new sounds I estimate that over 2000 are very useful. Unfortunately, it takes quite a few seconds to recall a sound, providing the Algorithm number and sound number can be remembered! Additionally, the synthesiser can hold another 32 sounds, and by using Algorithm changes we have a further 300 to 400 new sounds to add to the library! If these were available, stored on a £3 disk, it would be considerably cheaper than the 72 RAM packs required to store the same number. Remember that each one costs £70! And we haven't even started to create new sounds, but have merely modified existing ones! I said it was complicated, didn't I?

Once a new sound has been created we now come to the actual playing bit. The physical size of the keyboard covers five octaves, sixty notes, but the actual number of notes available is 127. Middle C on the piano is actually key number sixty on the DX7. When playing 'live' it takes a few seconds to switch in the next group of five octaves and a similar amount of time to switch in the last seven notes. In reality one is restricted to the five octaves selected. The computer, on the other hand, can access all of the notes smoothly and with great speed. This is another plus in favour of using a computer to assist you in

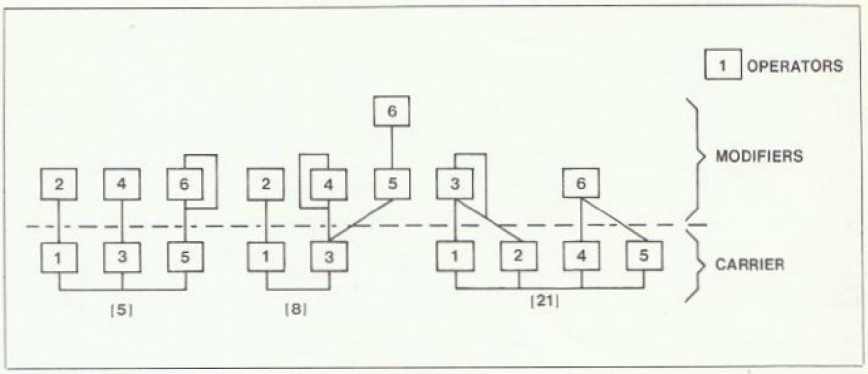


Figure 1.

playing. The DX7 is a Polyphonic Synthesiser which means it can play more than one note simultaneously. In fact sixteen notes can be played polyphonically. Indeed, the DX7 is one of the few that is capable of doing this. This opens up the possibility of the computer and artiste playing duets! By 'splitting' the keyboard, that is having one type of sound at one end of the keyboard and another sound at the other, some very interesting effects can be created. This opens up the whole system to exciting orchestrations and compositions. For example, the very beautiful piece of music composed for the organ by Widor, the Toccata in F, is well within the capabilities of the DX7 under MIDI control, with notes still available for further embelishments! Additional features such as 'pitch-bending', 'modulation control', 'swell', 'aftertouch' and 'breath-control' are still available, plus the ability to have fifteen similar instruments playing at the same time! The mind boggles when one actually stops and thinks about all the possibilities.

MIDI Language

It is now that we roll up our sleeves and look at the second topic, that of the actual language of MIDI. I will state here and now that this next section is not easy to grasp at first, but will take a couple of attempts before the essential features are taken in. Take your time and re-read as often as is necessary.

Because of the speed of data transfer (32.25 Kbaud) a great portion of any program will have to be written in Machine Code. Basic and even compiled Basic just is not fast enough. If you are keen to write your own programs I hope that you are following Keith Mayhew's excellent series on Machine Coding. The essential features of programming a 6502 are dissimilar to a 68000, there is a vast difference between these CPU's, but the knowledge and experience gained will be invaluable.

As with any language, MIDI uses words and sentences. A MIDI word is shown in Figure 2. As you can see, it is composed of ten bits. Each word is transmitted serially in 320 microseconds. The word is actually 8 bits long but two flags are required to signal the start and end of each word. That is logical, isn't it? The instruction is contained within bits D0 to D7. As you can see, each word is composed of a series of binary digits, a '0' or a '1'. There are two types of MIDI byte, the STATUS byte, which is identified by a 1 at the start of the word, i.e. 1001, and a DATA byte, which starts with a 0.

STATUS bytes are always the first word in a MIDI sentence and are commands. DATA bytes, on the other hand, qualify the STATUS byte. In the example 1001, the STATUS byte informs the keyboard that a note is to be played and the following DATA bytes inform the keyboard which note is to be played and the manner in which it is to be played, such as velocity and whether aftertouch is required.

There are five catagories of MIDI data. These are known as Channel, System Common, System Real Time, System Exclusive and System Reset. The protocol for all but one of these groups is common to all pieces of MIDI equipment. The exception is System Exclusive.

Each manufacturer allocates its own specific System Exclusive Code and usually they are common to all the instruments they manufacture. In other words, all Yamaha System Exclusive codes are the same for all Yamaha Instruments. Software for the DX7 is compatible with the DX1, DX5, DX9 and so on. Where additional features are incorporated into an instrument, these features would also be under System Exclusive codes. We will come back to System Exclusive later.

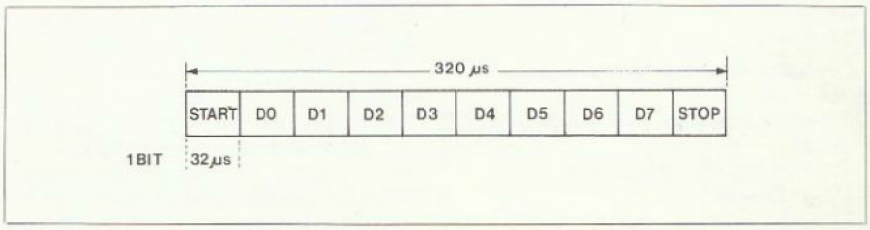


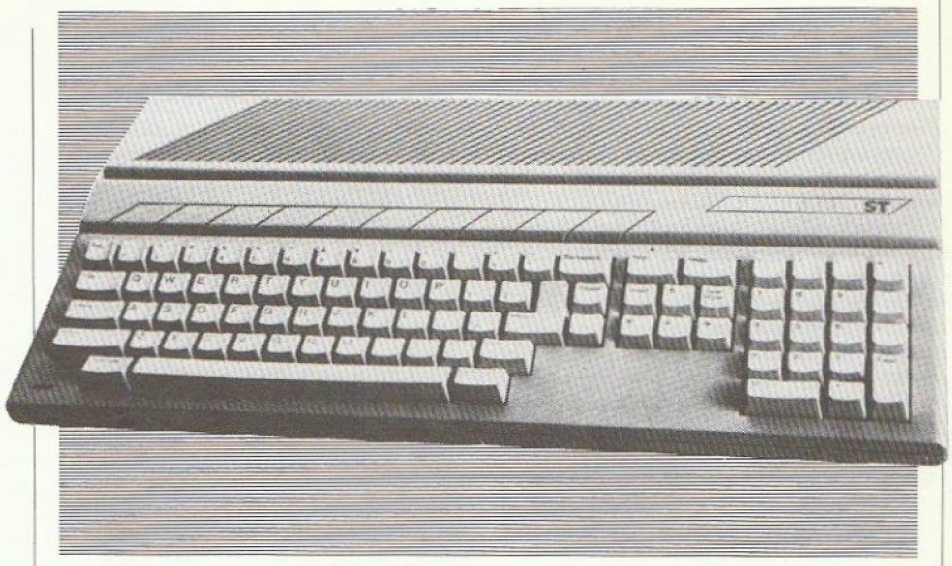
Figure 2.

1) CHANNEL.
1001nnnn KEY ON and channel number (nnnn=0000=channel 1)
0kkkkkkk KEY NUMBER (0 to 127)
0vvvvvvv KEY VELOCITY (0 = off 1=ppp 127=fff)
1000nnnn KEY OFF
1011nnnn CONTROL CHANGE
1100nnnn PROGRAM CHANGE
1101nnnn CHANNEL PRESSURE/ AFTERTOUCH
1110nnnn PITCHBEND
The Control Parameters on the DX7 are as follows:
1011ccccvvvv where c=1, Modulation wheel v=0 to 127
where c=2, Breath Control, "
where c=4, Foot Control, "
where c=6, Data entry knob, "
where c=64, Sustain switch, v=0 off, v=127 on
where c=65, Portamento switch " "
where c=96, Data entry +1 v=127 ON only
where c=97, Data entry −1 v=127 ON only
1100nnnn Program change and Channel number
0ppppppp Program number. Where p=1 to 32, Internal programs are selected; where p=32 to 63, Cartridge programs are selected.
1101nnnn After-touch and Channel number (n=0:Channel 1)
0vvvvvvv Touch value (0 to 127)
1110nnnn Pitchbend and Channel number (n=0:Channel 1)
0vvvvvvv Pitchbend value (LS byte)
0vvvvvvv Pitchbend value (MS byte)
When the MS byte is between 0 and 64, the LS byte is 0
When the MS byte is between 65 and 127, the LS byte is 2.

As you can see from this listing, the structure is quite complicated, but it is always presented in the same manner, irrespective of whether the code is being transmitted, or received. Each status byte will always have the correct number of following bytes. Remember that I am referring to the DX7 specifically, an instrument that does not have a touch-sensitive keyboard will not be able to transmit this data. It will be able to receive it, but it would be ignored. This table illustrates the compliment of relevent messages in Channel Information.

Status D7.....D0	No. of bytes included after
1000nnnn	2
1001nnnn	2
1010nnnn	2
1011nnnn	2
1100nnnn	1
1101nnnn	1
1110nnnn	2

A typical sequence of data might be the following:
1001nnnn + 0kkkkkkk + 0vvvvvvv (Note on)
10010000 00111100 01010000 (values in Binary)
144 60 80 (values in decimal)



The translation would be 'Play middle C on channel 1 with a velocity of 80'. To stop the note, another string of data would be transmitted as follows:
10000000 00111100 01010000 (Binary)
128 60 80 (decimal)

The 128 command stops the C on channel 1 with a release rate of 80.

This constitutes the basic commands, but of course the total length can encompass all the relevent commands within Channel Information.

Key Aftertouch, Channel Aftertouch, Control Change, Pitchbend and Program Change. One distinct advantage the 520ST will have over the popular micros like the BBC, Spectrum and Commodore, all of which have been very well supported for some years now with MIDI hardware and software, is the ability to use Aftertouch and Control changes freely. These features are notorious 'memory gobblers'. The memory available on the micros mentioned above is rapidly used up when these features are used. Obviously having half of a Megabyte of RAM available will allow free use of this very valuable form of expression with a system comprising a 520ST and a synthesiser such as the DX7. In fact after the beautiful sound quality of the DX7, the touch sensitive keyboard was the most important feature I liked!

I am digressing, back to Channel Information. nnnn indicates which MIDI channel is selected (1 to 16); kkkkkkk indicates which note is to be played (0 to 127) in this manner:
********* PIANO RANGE *********
C C C C C C C C C C C C (middle C=60)
0 12 24 36 48 60 72 84 96 108 120 127
vvvvvv denotes the value of the KEY velocity. It is a logarithmic value:
where 0 ----------64 ----------127
OFF ppp pp p mp mf f ff fff

The default value is 64 for those instruments that have no velocity sensing. When one considers that a number of different Controllers can be installed in an instrument, the only one that has a general coding is Pitchbend. All the other forms of control are assigned a coding by the manufacturer. Each instrument will have in its manual a table of control allocations. Controllers can be one of two types; an ON/OFF state (such as Sustain) or VARIABLE (such as Modulation, Breath-control, etc.). The Control Number, cccccc (0 to 127), is divided into logical sections. Where:
cccccc=1 − 31, this refers to Controller(s) 1 − 31 MSB
=33 − 63, " 33 − 63 LSB
=64 − 95, this refers to Switches (ON/OFF)
=96 − 124 are allocated by the Manufacturer
=124 refers to Local or Remote control of the Keyboard.
=125 all notes OFF command
=126 selects MONO mode
=127 selects POLY mode

Continuous Controllers are identified by the Most Significant byte and Least Siginificant byte where the resolution is greater than 7 bits; otherwise only the MS byte is sent. Where greater than 7 bit resolution is used, the MS byte is sent first followed by the LS byte.

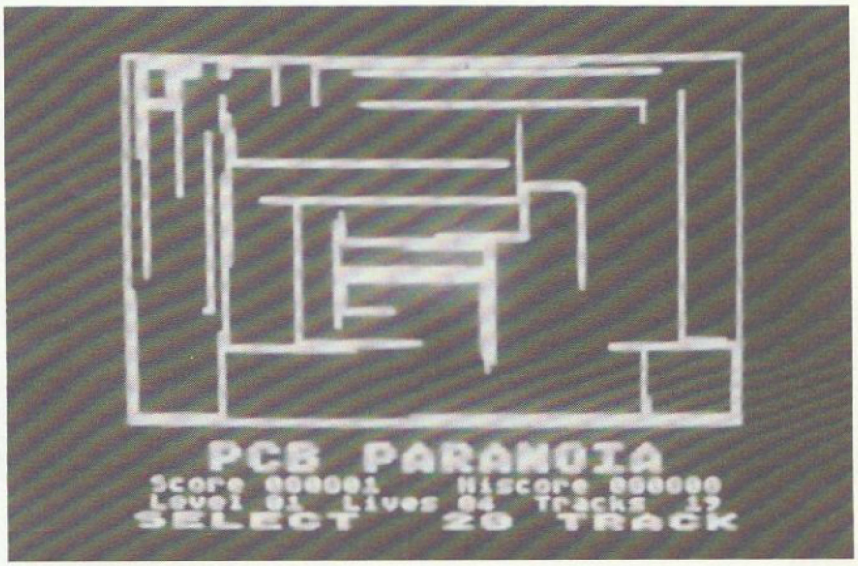
Often, only the LS byte is changed, in this instance only the LS byte is sent. The range is similiar to those already seen:
For CONTINUOUS Control:
0 64 127
min max
For SWITCHES
0 127
OFF ON
For PITCHBEND
0 64 127
LOW CENTRE HIGH

The sensitivity of the Pitchbend is selected in the receiver. An important feature to note is that a Dummy Byte is sent with all Notes OFF/Mode Select data in order that all data lengths are equal. This concludes the first part of MIDI Protocol.

In the next part of this series, I will complete the structure of the MIDI language and hopefully have some photographs and examples of MIDI in use.

PCB PARANOIA

by Mister X *Runs in 16K*



Typing It In

Listing 1 is the Atari Basic program that will create an auto-boot version of PCB PARANOIA on disk or cassette. Obviously, the DATA is in hexadecimal in order to make the program as small as possible.

Cassette Instructions

Type Listing 1 into your computer using BASIC. If you are not using KEYO just ignore the two checksum letters before the line numbers. When you have finished typing it in, save it off first, this will be your back-up copy in case things go wrong from here on, you will then be able to reload all your precious typing. Now RUN the program. You will be asked if you require a cassette or disk version. You should type 0 and press RETURN for cassette. The program will start checking the DATA, and will notify you of any errors. When all the DATA is correct, the computer will prompt you to insert a blank cassette into your recorder, press PLAY and RECORD and then press RETURN. The program will now save off an auto-boot cassette version of PCB PARANOIA.

To play PCB PARANOIA, turn off your computer, remove all cartridges and, pressing the START key, turn on the computer. (If you own an XL or XE then press OPTION as well.) Press PLAY on the recorder, and then RETURN. PCB PARANOIA will load and run automatically.

Disk Instructions

Type Listing 1 into your computer, using BASIC. If you are not using KEYO then just ignore the two checksum characters before each line number. When you've finished, save a copy to disk as a back-up then RUN the program. Type 1 and press RETURN to tell the program to make a disk version of PCB PARANOIA.

The program will check the DATA statements, notifying you of any errors. Once all errors have been corrected, the program will ask you to insert a disk with DOS and press RETURN. Use either DOS 2 or DOS 2.5. The program will now create an AUTORUN.SYS file on that disk. Do NOT change the filename!

To play PCB PARANOIA, insert the disk containing the AUTORUN.SYS file into your drive, remove all cartridges and boot the system. PCB PARANOIA will load and run automatically.

Playing PCB PARANOIA

Percy PCB designer works late one night. Percy starts to feel drowsy. Percy notices something strange. Percy looks at his artwork. Percy gets a shock. Percy sees the PCB tracks moving. They are alive! Percy is under attack from tracks. Percy fights back...

As Percy PCB Designer, you must defeat the hoards of evil rampant tracks lurking on your nice drawing board. Select the number of tracks you feel you can cope with using the SELECT key. Press START or the trigger of joystick 1 to start the game. Your yellow track will appear in the centre of the drawing board. You must avoid the lethal rampant red and blue tracks at all cost, forcing them into blind alleys, where, without paper to draw on, they will die.

Should you yourself get caught, you can press the trigger to explode the one bomb you have per screen which will clear your immediate vicinity. Extra lives and points are awarded as the game progresses. SPACE bar pauses the game, and any key will restart it. SELECT or OPTION will abort the current game.

Get going and destroy those tracks!

Listing 1.

```
MH 10 REM ** PCB PARANOIA **
RZ 15 REM BASIC LOADER FROM ANALOG
YT 20 TRAP 20:? "MAKE CASSETTE (0) OR DIS
   K (1)";:INPUT DSK:IF DSK>1 THEN 20
UK 30 TRAP 40000:DATA 0,1,2,3,4,5,6,7,8,9
   ,0,0,0,0,0,0,0,10,11,12,13,14,15
LI 40 DIM DAT$(91),HEX(22):FOR X=0 TO 22:
   READ N:HEX(X)=N:NEXT X:LINE=990:RESTOR
   E 1000:TRAP 120:? "CHECKING DATA"
JG 50 LINE=LINE+10:? "LINE ";LINE:READ DA
   T$:IF LEN(DAT$)<>90 THEN 220
UU 60 DATLINE=PEEK(183)+256*PEEK(184):IF
   DATLINE<>LINE THEN ? "LINE ";LINE;" MI
   SSING!":END
LY 70 FOR X=1 TO 89 STEP 2:D1=ASC(DAT$(X,
   X))-48:D2=ASC(DAT$(X+1,X+1))-48:BYTE=H
   EX(D1)*16+HEX(D2)
FG 80 IF PASS=2 THEN PUT #1,BYTE:NEXT X:R
   EAD CHKSUM:GOTO 50
BG 90 TOTAL=TOTAL+BYTE:IF TOTAL>999 THEN
   TOTAL=TOTAL-1000
LR 100 NEXT X:READ CHKSUM:IF TOTAL=CHKSUM
    THEN 50
MO 110 GOTO 220
ZR 120 IF PEEK(195)<>6 THEN 220
EQ 130 IF PASS=0 THEN 170
MY 140 IF  NOT DSK THEN 160
DB 150 PUT #1,224:PUT #1,2:PUT #1,225:PUT
    #1,2:PUT #1,112:PUT #1,32:CLOSE #1:EN
   D
IJ 160 CLOSE #1:END
IS 170 IF  NOT DSK THEN 200
GO 180 ? "INSERT DISK WITH DOS, PRESS RET
   URN";:DIM IN$(1):INPUT IN$:OPEN #1,8,0
   ,"D:AUTORUN.SYS"
YO 190 PUT #1,255:PUT #1,255:PUT #1,0:PUT
    #1,32:PUT #1,38:PUT #1,38:GOTO 210
SD 200 ? "READY CASSETTE AND PRESS RETURN
   ";:OPEN #1,8,128,"C:":RESTORE 230:FOR
   X=1 TO 40:READ N:PUT #1,N:NEXT X
QS 210 ? :? "WRITING FILE":PASS=2:LINE=99
   0:RESTORE 1000:TRAP 120:GOTO 50
MI 220 ? "BAD DATA: LINE ";LINE:END
RQ 230 DATA 0,13,216,31,255,31,169,0,141,
   68,2,169,60,141,2,211,169,9,141,231,2,
   133,14,169,38,141,232,2
JE 240 DATA 133,15,169,112,133,10,169,32,
   133,11,24,96
TE 1000 DATA 7070704D0030D04710200202064
   002F8080F0E3E280F0E1F2E1EEEFE9E1808000
   33636F7265001010101010,682
LK 1010 DATA 00000000286973636F7265001010
   1010101000002C6576656C001011000002C6976
   65730010150000347261636B,957
ZW 1020 DATA 730000121000332352C252334000
   1210003432212328A9148583A9008D08D28582
   200422A9038D0FD2206D2120,667
IV 1030 DATA D12020B721203325204E25A58385
   8FA90185A2206D2120852120B3232036222022
   2220DB2320C82220E8252026,221
FT 1040 DATA 22ADFC02C921D00BA9FF8DFC02AE
   FC02E8F0FAAD1FD0C907F0DB4C8320A000B900
   2099002FC8C00690F5A90DA2,130
```

Continued on page 31.

STARTING FROM BASICS

by Captain Hacker

Welcome to the third part of my series of articles on BASIC for the beginner. The last episode dealt with variables, and in this issue I will deal with how to control the flow of a program.

GOTO

It is almost impossible to write a Basic program without using the GOTO command, and in the last article in this series several of the programs used it to force our program to go back to a previous line number and execute a series of commands again – we created a loop. GOTO can in fact be used to force your program to continue executing at any line number, and it is this ability which often encourages programmers to write in a very untidy manner, with jumps all over the program. The result is an unnecessarily over-complicated piece of spaghetti which is almost impossible to de-bug. Budding programmers be warned – avoid over use of GOTO as your programs grow.

GOSUB

It is not always apparent to the beginner just how important and useful this command is. Suppose that, in your program, there is a particular function (or set of commands) that you might want to perform fairly regularly through out your program. Let's choose something simple as an example, perhaps you might have to print the value of several variables at various stages of your program. Let us suppose that we have two numeric variables, labeled A and B, whose values change as the program proceeds. Type in the following program.

```
10 A=29:B=13
20 PRINT "A=";A,"B=";B
30 PRINT "A+B=";A+B
40 PRINT "A*B=";A*B
50 A=A+45:B=B*34
60 PRINT "A=";A,"B=";B
70 PRINT "A+B=";A+B
80 PRINT "A*B=";A*B
90 A=A+62:B=B/16
100 PRINT "A=";A,"B=";B
110 PRINT "A+B=";A+B
120 PRINT "A*B=";A*B
999 STOP
```

RUN the program and you will get a list of numbers printed.

You can see that we have repeated the same print commands in three different positions in the program, surely there must be a more efficient method available? This is where the GOSUB comes in, for we can place the PRINT commands into another part of the program and make them a subroutine. Take a look at the following program:

```
10 A=29:B=13
20 GOSUB 20000
50 A=A+45:B=B*34
60 GOSUB 20000
90 A=A+62:B=B/16
100 GOSUB 20000
999 STOP
20000 PRINT "A=";A,"B=";B
20010 PRINT "A+B=";A+B
20020 PRINT "A*B=";A*B
20030 RETURN
```

Type RUN, and you will see that the second program gives exactly the same results, but with less lines, and a lot less typing. So how does the GOSUB command work? Initially it appears to operate just like a GOTO, since the statement GOSUB 20000 forces the computer to jump directly to line 20000, skipping over any lines in between, and continue running from there. The difference here, though is that the computer REMEMBERS where it came from, and when it encounters a RETURN command it goes straight back and continues as if it had not been diverted in the first place! The idea is of course that you try and break your program down and separate any regularly performed function, turning them into subroutines. It is usual to place subroutines at the end of a program, and this is why I have placed our example routine at line number 20000.

There is something important which you should remember when writing subroutines, always have just one RETURN command, (place it at the end of your routine to help clarity), and make sure
that this is the ONLY way that you leave the routine! It is not just considered sloppy programming to exit from a subroutine using the GOTO command, but it also wastes memory. Wastes Memory! How? I here you gasp. Well, it goes like this – evey time BASIC encounters a GOSUB command it has to make a note of and remember where it is before it goes to the subroutine in question. It takes 4 bytes of memory to store this, so BASIC helps itself to a little of your free memory to do so. As soon as BASIC encounters the RETURN command at the end of your subroutine, it picks up these four bytes so that it knows where to return to, and gives the equivalent memory back to your free memory area.

It is possible, however, to perform something called NESTING of subroutines, where, in the middle of one subroutine you may have a GOSUB calling another, and so on. Now, so that BASIC can find its way back properly from each RETURN command, it must store ANOTHER four bytes away for each nested subroutine, and it does this using something called a STACK. You will come across this term quite regularly if you take up programming seriously (especially if you progress to machine code), but it simply means a stack for numbers, where numbers are added to the top as necessary, and when they are taken back they are pulled of the top so that the last number added to a stack will always be the first one removed, or PULLED, to use the usual term. BASIC creates this stack, and it will expand and contract it as it encounters GOSUB and RETURN statements, and consequently the more deeply you nest subroutines, the more the stack will grow, and the more you will lose from your free memory space.

This temporary loss of memory space is not usually a problem, indeed many programmers are probably not even aware that it happens, until they start commiting the heinous crime of using the GOTO command rather than RETURN to exit from a subroutine! The following short program will illustrate this point better than a thousand words of explanation, so type

NEW, enter the program into your computer, and then type RUN. (All the variable names are fictitious and bear no resemblance to any program living or dead).

```
10 A=1
20 A=A+1
30 GOSUB 21000
40 GOTO 20
21000 B=A
21030 GOTO 20
21090 RETURN
```

You should notice that it runs quite happily for a few minutes, and then suddenly crashes with the message ERROR 2 at line 30. The reason is, of course, that at line 30 we have given the command GOSUB 21000 to call the subroutine starting at that line number, but instead of going back via the RETURN command on line 21090, we have forced BASIC back to the main program with the GOTO 30 on line 21020. Consequently, BASIC thinks it is still in its subroutine and so has not removed the four bytes from the return stack. It repeats the whole process again and again, each time increasing the size of the stack until the stack occupies virtualy all of your memory, at which point of course, it crashes.

You can actually watch your memory being eaten away before your very eyes, if you wish by including the following line.

```
21010 PRINT FRE(0)
```

Type RUN, and you will witness the trick of the disappearing memory! FRE(0) is of course the function used to see how many bytes of free memory you have.

BASIC does actually have a command to remove four bytes from the return stack without using the RETURN command; it is called POP. This is a disgusting command since it only encourages the kind of perverse programming that I have just illustrated, so I hereby BAN all of you beginners from using it. (I fear that it is probably too late to save many unfortunate sinners who have long since strayed down this path leading to DOOM, DESPAIR, DESPONDENCY, SPAGHETTI JUNCTION, endless cups of strong coffee and many, many late nights trying to locate elusive BUGS). I therefore positively refuse to give you any examples of its use! (NOTE FROM EDITOR – try including the following line:
21020 POP that should do the trick).

The only decent, law abiding way to correct this program is of course to remove line 21030 and allow the RETURN command line 21090 to take BASIC back to the main program flow. Well, you have been WARNED – keep it neat, or face the consequences!

Before we leave the GOSUB command try this dynamic little one liner:

```
10 GOSUB 10
```

IF....THEN (IT'S A TOUGH DECISION!)

It's about time your computer made it's first decision in life, and so this is where we meet the IF command.

Very few programmers actually appreciate exactly how the IF-THEN statement really works, and in fact many have come to assume that it exists to make comparisons itself. This is simply not true, for value comparisons are a separate function in their own right. If you are a beginner the last sentence may not make much sense, but please bear with me for all will be revealed in time! For those of you who believe yourselves to be hardened hackers, try and forget everything you thought you knew about the IF-THEN command and read the following paragraphs carefully, for you may well find you can improve your programming technique dramatically.

The IF-THEN command is merely a way of testing a number and deciding whether it has either a value or is zero. If the number (or variable) placed between the IF and THEN does not have a value then the rest of the statement is ignored completely, and BASIC looks for the next line number. If, however, the number (or variable) has some value, regardless of what it actually is, BASIC will continue with the line and execute any commands you have placed after the THEN part.
To demonstrate what I mean, type the following line:

```
IF 0 THEN PRINT "YES"
```

As is expected, the computer does nothing. It examines the number between the IF and THEN, and since its value is zero it aborts the line. Now try this line:

```
IF 1 THEN PRINT "YES"
```

This time, of course, the computer's response is a little more positive – it now prints the 'YES'! The value between the IF and THEN does not actually have to be a 1 to qualify, in fact any value, mathematical expression, or variable with a number other than zero will allow BASIC to continue past the THEN.

To illustrate this fact try entering each of the following commands:

```
IF 10 THEN PRINT "YES"
IF 196735 THEN PRINT "YES"
IF 0.3 THEN PRINT "YES"
IF 0.00001 THEN PRINT "YES"
IF -6 THEN PRINT "YES"
```

Each of the above will give the result 'YES' since each of the values between the IF and THEN are said to be 'TRUE'. A value of zero is said to be 'FALSE'.

When we use the IF-THEN statement in a program, however, it is more usual to use it to test the contents of variables, or even the results of mathematical expressions. For example, the following set of lines set up a variable and then test its value in several ways. Type each of them into your computer in the order in which they are presented:

```
A=5
IF A THEN PRINT "YES"
IF A-2 THEN PRINT "YES"
IF A-5 THEN PRINT "YES"
IF A/2 THEN PRINT "YES"
```

Notice that the only IF-THEN line that does not give a TRUE response is the one with the expression A-5 since it is the only one where the expression evaluates to zero. There is a way in which you can make calculations more clear when programming, and this is to use the brackets, (). You can separate, or make an expression stand out as a section, by using these, as in the following example:

```
IF (A-5) THEN PRINT "YES"
```

The brackets effectively mean calculate the contents of the brackets, and then apply this resulting value to the rest of the line. To illustrate this point further, try the following lines in sequence:

```
A=5
IF 6*(A-5) THEN PRINT "YES"
IF 6*A-5 THEN PRINT "YES"
```

On the face of it, the two IF-THEN lines should have the same result, but this is not so. What we have done with the brackets is to alter the sequence in which BASIC makes the calculations. With the expression 6*(A-5), BASIC calculates the A-5 part first, and then applies the result (zero) to the rest of the calculation. We are of course not limited to just using the PRINT command with our IF-THEN statement; we can actually use any of the BASIC commands, even another IF-THEN sequence! The most popular, though is the GOTO. Here is a typical example of the use of the IF-THEN:

```
10 A=10
20 PRINT A
30 A=A-1
40 IF A THEN GOTO 20
50 PRINT "END"
```

This little example program will do something completely useless, it simply prints the numbers 10 to 1 on the screen, but it does serve to demonstrate a CONDITIONAL LOOP – a loop which has a definite exit condition. Now enter and run the following routine:

```
10 A=1
20 PRINT A
30 A=A+1
40 IF(A-11)THEN GOTO 20
50 PRINT "END"
```

This one will print the contents of A, as A increases, until A–11 becomes false (zero in value), and this happens when A equals 11 itself. Something which is worth remembering is that on line 40 you can in fact omit the GOTO command and instead have the following:

```
40 IF (A-11) THEN 20
```

This is allowed because BASIC sees the line number (line 20) after the THEN, and it just assumes that a GOTO is intended. The brackets around the A–11 are not necessary either, but I have included them for the sake of clarity. Experiment with the IF-THEN command, it is important that you become familiar with it since it is the major decision-making instrument available to the programmer.

COMPARISONS

It is often necessary to be able to compare two numbers, or strings, and know whether they are equal, or perhaps which is the greater. To do this we use the comparison symbols, which are =,<,>and < >. Lets take a close look at these.

A little Equality

Suppose that we want to compare two numbers to see if they are equal in value. We would accomplish this with the aid of the = sign, i.e. 1=1. Now, the result of a camparison is of course going to be one of two things, it can be either TRUE or FALSE. The computer gives us the answer in the form of a number – 0 for FALSE or 1 for TRUE, and we can use this number in calculations, further comparisons or even just print it on the screen. Type the following line, and you will see what I mean:

```
PRINT 3=4
```

BASIC will compare the numbers 3 and 4 to see if they are equal in value. Since they are not, the expression gives a FALSE value, or zero, which is then printed to the screen. As I mentioned earlier, it is often a good idea to use brackets to separate expressions, so you might find it clearer to try the following:

```
PRINT (3=4)
```

This again prints 0 to the screen. Now try the following:

```
PRINT (3=3)
```

This time the result of our expression is TRUE, so the computer prints 1 to the screen.

Comparing literal numbers like this is of course pretty useless, since we know that 3 does not equal 4, and that 3 does equal 3. But suppose however, that we have two variables A and B in our program whose values are constantly changing, and that at various stages we want to see whether they are equal. We can of course use the same expression but with variables in place of literal numbers i.e.

```
A=3:B=4
PRINT (A=B)
B=3
PRINT (A=B)
```

Greater or Lesser

It is often desirable to know if a variable is greater or lesser than a certain value or other variable. To do this we use the angle brackets, < or >, as in the following example:

```
PRINT (4>3)
```

In the above example we are testing to see if 4 is greater than 3, which of course returns a TRUE value, 1. We can express this another way by saying that 3 is less than 4, i.e:

```
PRINT (3<4)
```

This once again gives TRUE. If, however, we were to test to see if 3 is greater than 4 the result would naturally be a FALSE, i.e:

```
PRINT (3>4)
```

Notice that the important point here is which way round you place the angle bracket; the number facing its WIDE side is compared to see if it is greater than the number facing its thin side, and visa-versa.

To demonstrate the use of this comparison more clearly, here is another mind-bogglingly useless program for you to enter. When you RUN it, it will keep asking for a number to be typed, so try entering random numbers between 0 and 10. Notice that it will print 0 for numbers 5 or less and 1 for numbers greater than 5.

```
10 INPUT A
20 B=(A>5)
30 PRINT B
40 GOTO 10
```

It is also possible to combine some of the comparison signs to alter the way in which they operate. For example, < > means NOT EQUAL TO, i.e. if two compared values are NOT equal then a TRUE is returned. We can also use the two expressions >= or <=, which means GREATER THAN OR EQUAL TO, and LESS THAN OR EQUAL TO respectively. Try the following examples, and experiment with some of your own:

```
PRINT (4<>5)
PRINT (4<>4)
PRINT (3>=3)
PRINT (2>=3)
PRINT (6<=6)
PRINT (7<=6)
PRINT (3<=6)
```

Comparing Strings

Just as we are able to compare numbers and number variables, so we are also able to compare strings and string variables. Try the following two examples:

```
PRINT ("HELLO"="BYE")
PRINT ("HELLO"="HELLO")
```

Notice that the same rules apply if the comparison is TRUE, BASIC returns 1, but if it is FALSE 0 is returned.

We can also compare two strings to see if one is greater than the other, but here we are not testing the length of the string. We are testing its alphabetic value, or, if you like, how far through the dictionary the words would appear – the closer a word is to 'A', the lower it's value. For example:

```
PRINT ("XXX">"AAAAA")
```

This will give a TRUE, since BASIC considers that 'XXX' is greater than 'AAAAA'.

In Combination

Well, now you have seen how the IF-THEN function operates, and you have also seen how the comparison functions work. On their own, these two facilities are of little use, but together they become one of the most valuable commands available to the BASIC programmer.

Here are a series of examples to show how these two facilities are often used together – note, though, I have not always included the brackets, since they are not usually used on simple comparisons inside IF-THEN commands. Do not bother typing these in, they are just abstract lines that you might find in a program:

```
IF A=4 THEN PRINT "FOUR"
IF A=B THEN PRINT "SAME"
IF B>C THEN GOTO 560
IF B>C THEN 560
IF (A$="YES") THEN A=A*2
IF B$<>"NO" THEN 600
IF (A$<B$) THEN C$=B$
```

IN CONCLUSION

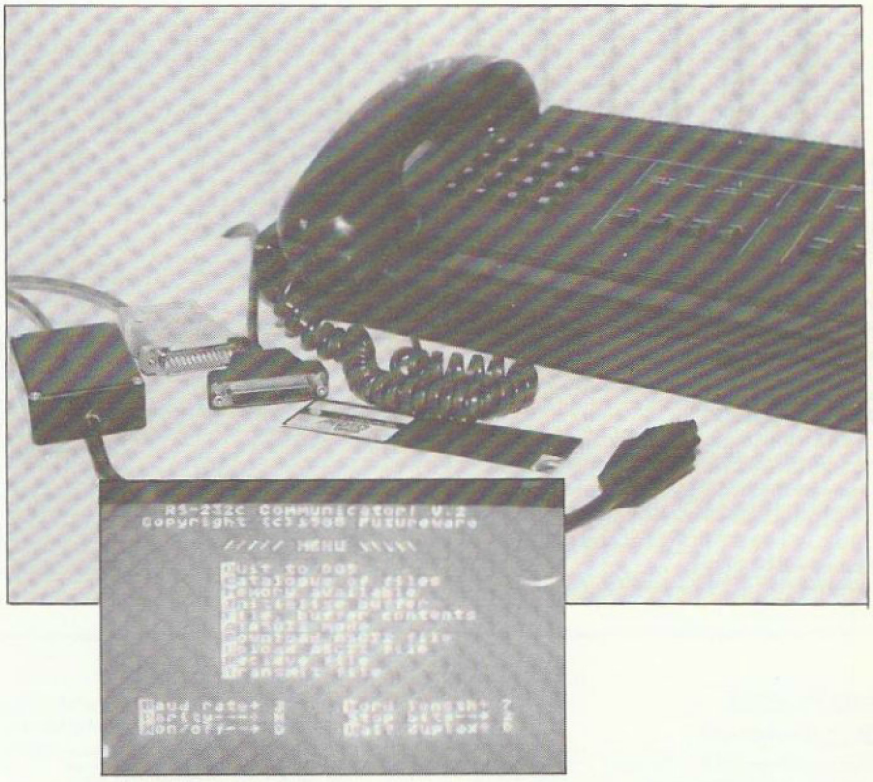
As a final example of the use of IF-THEN and comparison functions, here is a version of the old Higher-or-Lower number guessing game. Don't worry about the new functions on line 10, I will cover these in a future issue!

You should experiment with the commands discussed in this issue since they are the most important program-flow control functions available to you in BASIC. In the next part of this series we shall investigate further ways of controlling our program flow.....See you soon.

```
10 A=INT(RND(0)*100):C=0
20 PRINT "NEW NUMBER SELECTED."
30 PRINT "WHAT IS YOUR GUESS....";
40 INPUT B:C=C+1
50 IF B<A THEN PRINT
"TOO LOW":GOTO 30
60 IF B>A THEN PRINT
"TOO HIGH":GOTO 30
70 PRINT "CORRECT! IN ";C;" MOVES!"
80 GOTO 10
```

Beginning in the Comms World

by Matthew Tydeman



Telecommunications was virtually unheard of in England 3 years ago. Many people could just about afford a computer, let alone a Modem and other equipment needed for entering the world of Telecommunications (Comms). Today, not only has the price of computers dropped, but so has the overall price of Modem hardware and software. But most visibly, the hardware is now readily available. American Atari owners have had the priviledge of being in the world of Communications now for many a year. Us Brits however have only had a few years to actually enter and explore this fascinating world. The Minor Miracle Modem has changed the Atari Comms scene drastically over these past years. This Modem complete with software and all necessary interfacing made Comms readily available to anyone who was interested, at a reasonable price, tempting every possible Atari owner.

The WS2000 Modem (Minor Miracle) isn't the only Modem available to Atari owners. Virtually any Modem can be easily connected to your Atari with the aid of specialised interfacing. If you are interested in purchasing a Modem but have no idea were to turn to decide what equipment you should acquire, read on – there are a lot of options open to you.

The WS2000 Modem is compact and very efficient. Its price and support obviously attract the Atari owner, which is understandable, but what can this Modem do that others can or cannot do? The WS2000 runs at baud rates (The speed at which data is transfered) 75-1200 allowing great flexibility in the system you log onto when calling BBS's and talking to the outside world. The faster it can operate the less time you'll be on the phone! At £199.95 it comes complete with the Viewterm software which allows a variety of configurations to be setup. The software can handle a vast possibility of operation making it quite a good piece of software. Although it is quite easy to use and complete, I find it rather lacking on the additional useful options which I've grown to love with other terminal software. The Modem has to be connected to the standard Atari serial port which can be found on the side of your computer in which a cassette, disk drive or printer may already be plugged. The interface with the WS2000 does this for you – you simply plug the interface in at the Modem end and the other into your computer and you're away. Also available is the British Telecom Datelphone 1200, another Modem I have managed to test which in my eyes comes out on tops and you'll soon see why! This Modem is the one I'm currently using but its a bit of a hefty wack at the old bank account, at around £1200. Its not cheap but well worth the extra money. The Modem itself doubles as another phone as it has its own built in handset. Equipped with such wonderful features as Auto Answer, Auto Dial and even a self test to check operations are running OK. The The Modem has the possibilty of running at 9600 baud when using two Futureware Comm/Cables (described below) or two machines with built-in RS232, but normally has a standard transmitable baud rate of 1200/1200 along the phone line.

Dialling can be made from the keyboard or from the phone itself with the use of the push button keys, while the handset remains down but open. This Modem can connect to the now, very hard to come by 850 interface, but alternatively can be used with the WS2000 interface or the interface I tried from a new company, Futureware of London. They manufacture such a product at a very cheap price (£39.95) which includes an amazing terminal program with a higher spec than the Viewterm software. Communicator!, as it is called, is programmed in Assembly Language by J. Sosta and is cleverly done at that. Its very simple to use and very full featured down to the last possible configuration. The interface comes with its own handler which is best used with its companion program, Communicator!.

Having used both Modems I found the BT Datelphone coupled with the Futureware Comm/Cable package the best to use. I don't think the WS2000 Modem had enough features for possible expansion of my system at a later date, whereas the BT model and Futureware software will handle all my needs. If you are a beginner in the Comms world and really only want to use your Modem to talk and leave messages, then the WS2000 is fine and at the price offered you can't go wrong. I suggest, for beginners and those who eventually want to move forward, that the Futureware software/hardware is possibly the best buy in this particular area today.

Listed here are a few BBS phone numbers to get you started. Note that most BBS's run at 300 baud set at 8 bit word, No parity and 1 stop bit. Some use a 'ring back' procedure which means that when you have dialled the number you should let it ring once and then hang up, then dial again and let the phone ring until you get the tone, then switch your Modem into operation.

Bulletin Boards

LBBS 0506 38562 24Hrs
BABBS 0225 23276 9pm-8am Weekdays
(ring back) 9pm-12am Weekends
ITEC 0268 25122 24Hrs
SABBS 0698 884804 24Hrs

Communications is a great world to enter, much fun can be had, meeting and talking to other people. Its something everyone likes and is now, thanks to many companies, readily accessible. Seriously consider it. It is the future.

Two-in One Competition

Monitor is proud to present not one, but two competitions in one. Take a look at the photo, study it well and then have a go at the competitions. You can enter just one, or you can have a try at both. Each competition carries a £10 prize plus a free 4 issues subscription to Monitor.
Send your answers to: Monitor, (2 in 1 Comp), P.O.Box 3, Rayleigh, Essex. Don't forget to include your name and address. Answers must be received by 10th January 1986.

Caption It

Look at the photo carefully. What do you think is being said or thought? Write a suitable caption that will impress the judges and you could be the lucky winner.

Guess Who?

Do you know who it is in the photo. No, not the Robot, the other one! Just put a name to the face and say what was his connection with Atari. All the correct entries will be put in the hat and a lucky winner will be drawn.

New Product Denouncements

Reprinted from ABACUS, the newsletter of the Bay Area Computer Users Society, USA.

Commode Business Machines today announced a new peripheral product for the Amigo computer. The Amigo 300 'Expansion Subsystem' is a complete unit that allows Amigo users to run a wide variety of IBM 370 mainframe software by performing IBM 3090 emulation. The compact 24' long, 6' high, 6' wide unit contains 128MB RAM memory, the processor complex, the 3092 Processor Controller, two 3370 800MB system hard disk units, and two 3097 power and coolant supply units. The 300 subsystem permits connection of the following IBM peripherals (or compatible equivalents): Ninety 3278 display terminals, sixteen 3430 tape drive units, twelve 3380 2520MB hard disk drives, six 3880 disk drive controllers, three 4245 200 lines-per-minute printers and a partridge in a pear tree! The 300 will run IBM system software including MVS/XA, VM/XA, JES2, JES3, UNIX, IMS DB/DC, CICs, PL/1, APL, RPG, TSO, SPF, VTAM, VSAM, PDF, PDQ, LBJ, and LSMFT. The subsystem connects to the Amigo CPU module through the expansion bus and includes software which turns the Amigo CPU, monitor and keyboard into an intelligent system console.

Commode Amigo spokesman Rick Geigerkounter claimed the 300 was developed to allow the ultimate in expandabilty and power for the personal computer user and "to enable Commode to crush the competition in a single blow". He indicated that the unit should be ready in time for Christmas. Price was not announced but several industry analysts estimated that it would not exceed £14,000,000.

In an unrelated announcement, Geigerkounter said that version 31.415927 of the Amigo operating system should be released next week.

Electronic Arfs, creators of the Pinko Destruction Set and the Muzak Construction Set, announced development of the Drywall Construction Set for the Commode Amigo computer. Flip Gawkins, President of Electronic Arfs, said that the system was aimed at Amigo owners who found their expanding computer system could no longer be contained in their present living or working space.

"The system supports windowing and dooring", claimed Gawkins. "With our optional foundation and roof-truss modules, it can support flooring and roofing. The Set will be particularly useful to purchasers of the Amigo 300 expansion unit who need some place to put it."

The basic Drywall set contains 50 4' by 8' gypsum wallboard panels, 1500' of 2-by-4s, 20lb of nails, 1500' of drywall tape and a hammer. The package contains CAD/CAM software which allows purchasers to configure their new computer rooms to suit their needs. Gawkins stated that the new product would be shipped to Amigo dealers within two weeks and that both of them would be carrying it. The system will also be available from lumberyards everywhere. Gawkins final comment was "I think this proves we are behind the Amigo 1000%!".

Modification to DOS 2.5

by Gordon L. Banks
Reprinted from Huntsville AUG Newsletter

With the much welcome release of DOS 2.5 comes the much welcome release from DOS 3.0 and its user unfriendliness. Still, as a staunch advocate of good old reliable DOS 2.0, I was a bit skeptical about giving up the modified version of DOS 2.0 to which I had become accustomed. Well, happily, it appears that my favourite modification works equally well on DOS 2.5.

If you wish to incorporate the mod into your system simply enter the POKE's and then rewrite your DOS.SYS file. This can be achieved quite easily in BASIC by entering OPEN#1,8,0,"D1:DOS.SYS": CLOSE#1 and press RETURN.

My modification is as follows: POKE 5903,65:POKE 5904,42: POKE 5905,46:POKE 5906,83: POKE 5907,89: POKE 5908,83: POKE 5909,155. This change alters the name of recognised AUTORUN.SYS files from just the one name (AUTORUN.SYS) to Axxxxxxx.SYS, where you can use any characters you like in place of the 'x's. For example, you might name your renumber utility ARENUMBR.SYS, or your screen-dump autorun as ASCRNDMP. SYS, or a little file that autoruns your disk menu as ARUNMENU. SYS. Now you can have as many autorun files on one disk as you like. When you boot up the disk, DOS will load and run the first file it finds that begins with 'A' and ends with the '.SYS' extender. Rename the ones you wish DOS to ignore by using the '.ARF' extender. This reminds you that they are AutoRun Files.

800XL Ramdisk

Reprinted from Austin ACE, USA.

The first step is to boot up in Basic with DOS 2.5. Then type IF PEEK(1802) < 182 THEN POKE 1802,PEEK(1802)+128 followed by pressing RETURN. Next press RESET and then call DOS. Check the directory for D8, it should say 000 free sectors. Format D8 and check the directory again, this time it should say 499 sectors, but there are really only 128 sectors available. If you try to use more than 128, the computer will crash. Now you can go to Basic and use your Ramdisk as D8:, or you can write DOS files to D8, then delete DOS.SYS leaving DUP.SYS. If you now go to Basic and POKE 5439,56 then return to DOS and create MEM.SAV on D8. Return to Basic and now when you call DOS, MEM.SAV and DUP.SYS will work from the Ramdisk. You will get the DOS menu almost instantly and your program will not be lost. In addition you still have 41 sectors on the Ramdisk in which to save programs. A point of interest: location 5439 tells DOS which drive to access for DUP.SYS and MEM.SAV. A 49 indicates drive 1, a 50 is drive 2, upto 56 for drive 8.

New Miracle Modem Range

A new range of professional modems from Miracle Technology have been released with 'Green Sticker' approval. There are 3 models in the WS3000 range and all feature auto-answer, auto-dial, speed buffered RS232 port, a control port and a 60 number quick-dial internal phone number directory. They are all approved for both the European CCITT and American BELL standards, and in addition are compatible with American Hayes command protocols.

The V2123 model runs 75/300/1200 bps, half duplex, CCITT standards V21, V23 and BELL 103, is upgradeable to 1200/2400 bps full duplex CCITT V22 and V22bis, price £295 plus VAT. The V22 adds 1200 bps full duplex to the V1232 specification and costs £495 plus VAT. The top range model is the V22bis which gives 2400 bps full duplex and costs £650 plus VAT.

ST Software

KUMA have released K-SEKA for the 520ST, it is 68000 assembler using standard Motorola mnemonics. It produces either absolute or relocatable code at a rate of 30,000 lines per minute. All functions can be carried out without accessing the disk. The main features are a Text Editor, Full 68000 Assembler, Symbolic Debugger, Line Disassembler, Built-in Linker, Conditional Assembly, Macro Facility, Formatted Listing Output, and Absolute, Relocatable or Linkable Code. All inclusive price is just £49.95.

Other products for future release are: K-SPREAD, K-DATA, K-WORD and K-COMM, and a book by John Braga called 'The Atari ST Explored'.

The Worm Has Turned

The long awaited third part of the Silicon Dream trilogy, called the 'Worm in Paradise' is now available on cassette for £9.95. In paradise every entertainment is to hand in the arcades and clubs. Everything is privatised, even the police force. This is a political SF adventure up to the high standard we expect from Level 9 these days. Graphics are included to enhanse the feel of the program. Multi-tasking ensures that there is no waiting whilst pictures are drawn, and the player can get on with the game. It has a vocabulary of over 1000 words and an advanced parser which allows long sentences to be input. Its quite possible that this will be Level 9's best yet!

1 Megabyte ST

Its not only software thats being made for the ST range. An additional 512K of internal system memory is now available from Softwerx, P.O. Box 71118, Murray, UT 84107, USA. The 'RamWerx 1040' is an upgrade board for the 520ST and included is a ramdisk utility so that immediate use can be made of the extra memory. It is claimed that applications running on the ramdisk are 5 to 50 times faster than on floppy disk. Connecting in the 1040 is only a matter of tacking in a handful of wires. After that additional boards could be added up to a total of 4 megabytes! RamWerx 1040 costs $275.

Other ST add-ons from Softwerx include: ShareWerx 360, a 5 ¼ inch drive that works like an ST drive and is IBM PC data and diskette compatible; ShareWerx 53, a special cable that lets you plug your 520ST 3.5 inch drive into an IBM PC and read 520ST disks; RamWerx 1500, which plugs into RamWerx 1040 for a total of 1.5 megabytes.

Free Poster

Level 9 are offering a full colour poster of their latest game, 'The Worm in Paradise', free to anyone who sends a large stamped addressed envelope to: Level 9, 'Worm in Paradise' Poster, P.O. Box 39, Weston Super Mare, Avon, BS24 9UR. The offer is while stocks last, so if you hurry you may still get one. (Readers outside the U.K. should send International Reply Coupons inplace of stamps.)

Pirates Rewarded

Computer Support are offering a £100 reward on prosecution of anyone involved in pirating of their products, plus they will replace an illegal copy with an original one, so that you can continue to use their excellent programs, but legally! Ring Computer Support on 01-311-7339, your call will be completely confidential and help stamp out piracy.

Corrigenda

Issue 8.

In Table 1 of Cracking the Code Part 4, please insert the opcode 'F9' into the line for 'SBC' under 'Absolute,Y' in the Arithmetic Operations.

PCB Paranoia *Continued from page 25.*

```
QF 1050 DATA 4F99002FC8CAD0F9BD062099002F
   C8E8E00A90F4A92D8D2F02A9008DC802A9968D
   C602A9388DC402A9258D0102,239
GD 1060 DATA A9C18D0002A928BD07D4A9038D1D
   D0A9018D6F02A9008597F8A202B599D59C900E
   D003CA10F5A202B599959CCA,692
SO 1070 DATA 10F9D8A000A202B59C2024259939
   20A59F993A20C8C8CA10EEA907A224A046205C
   E4A92F8D3102A9008D3002A9,497
SI 1080 DATA C08D0ED460A9308581A9008580A0
   00A20A989180C8D0FBE681CAD0F660A21FA9FF
   9D00309DE039CA10F7A93085,663
UP 1090 DATA 81A9208580A24EA000A9C09180A0
   1FA9039180A5801869208580900 2E681CAD0E6
   60202622AD10D0F03FAD1FD0,96
LZ 1100 DATA C906F02DC905D0EDE682A582C905
   9002A9008582AABD1322858385 8F20B3232004
   228D1FD0202622AD1FD0C905,48
BN 1110 DATA F0F9D0C4202622AD1FD0C906F0F9
   60AD10D0F0FB60A682BD18228D6820BD1D228D
   67206014283C506412141618,61
SX 1120 DATA 10000000011A232D002A205202F
   22CAD0FA60A514C514F0FC60A683CAAD0AD209
   03C978B0F79D0926AD0AD209,796
RW 1130 DATA 03C946B0F79D6E26BD0926C93590
   1FC949B01BBD6E26C91D9014C9318B102C0AD2
   300518691 4D00338E9149D6E,366
QE 1140 DATA 26AD0AD29D9C27290 31869019DD3
   26CA10B1A93F8585A9278586A9018587A90F85
   90608480A90006802A06802A,873
GW 1150 DATA 06802A06802A06802A1869308581
   8A4A4AA88A2903AAB1806080200802401004 01
   C0300C03A5838588A000B9D3,678
WX 1160 DATA 26306AA9048589AD0AD2D0032046
   23B90926858AB96E26858BB99C278584B9D326
   C901D002E68AC902D002E68B,803
VW 1170 DATA C903D002C68AC904D002C68B848C
   A48BA68A2099223DC422F00CA48CC689F05620
   46234CDF22B1801DC0222484,109
GL 1180 DATA 30031DC4229180A48CA58A990926
   A58B996E26C8C688F0034CCE2260A589C90390
   05AD0AD2858D248D300EB9D3,250
IK 1190 DATA 2638E901D002A90499D32660B9D3
   2618690 1C905D002A90199D32660A98099D326
   E68EA68E989D3727C68F20B3,441
FR 1200 DATA 23A200A00120D125A48CA58FD0AB
   6868A697A00020D125A900858E207625208D25
   A5972901D005E6982068254C,74
HJ 1210 DATA 8F20A58FC9649006A211A900F002
   A2008E5C20A200C90A9005E90AE8D0F709108D
   5E208A09108D5D2060AD00D3,679
AD 1220 DATA 290FC90FF0080590C90FF0028587
   A487982908D006E685A904D01F982904D006C6
   85A908D01498290 2D006E686,483
QW 1230 DATA A901D009982901D006C686A90285
   90A486A6852099223DC422D008B1801DBC2291
   80606868A900858E209525C6,374
QK 1240 DATA 98206825A598F0034C92204C8320
   A592854DD020A68EF050BC37278496C68EB909
   2618693C8D00D0B96E261869,277
ZL 1250 DATA 0C8591A9048592A491C692A6928A
   0A0A09308DC002290F09808D01D2A9328D00D2
   BDA324AAA9088593BDA72499,209
GE 1260 DATA 002AC8E8C693D0F4A592C901F027
   4C62E4000810180000000000000000000183C
   3C1800000183C7E7E3C1800,310
OX 1270 DATA 183C7FFFFF7E3C18A496B90926C9
   08B002A908C9779002A977AAB96E26C909B002
   A909C9479002A94738E908A8,297
WH 1280 DATA A58048A5814820 9922A200A90091
   80E004900CE00CB008889180C8C8918088A580
   1869208580 9002E681E8E010,565
QT 1290 DATA 90DC6885816885804CA0244829 0F
   0910859F684A4A4A4A091060A900858E859985
   9A859BA910A2059D2720CA10,1
FH 1300 DATA FAA90585984C6825F8A597C963F0
   111869018597D82024258D4720A59F8D4820D8
   60A59820 24258D5120A59F8D,49
XV 1310 DATA 522060A202A000B5992024259927
   20A59F992820C8C8CA10EE60A2FF86A086A1D0
   07A20086A0E886A1A908A204,294
KW 1320 DATA A4A08D1FD0490888D0FDA4A08D1F
   D0490888D0FDCAD0E9A5A11865A085A0D0DE60
   48A9008D0AD48D18D0A9288D,458
HU 1330 DATA 17D06840F89818659985998A659A
   859AA59B6900859BD82076256 0AD10D0D01BA5
   A2F017A90085A2A5858D6D26,985
VZ 1340 DATA A5868DD226E68EA68EA9649D3727
   6000000000000000000000000000000000000000
   0000000000000000000000,969
```

MONITOR ON DISK

Like the look of a program but can't find time to key it in? You've asked the wife three times to do it for you whilst you're out at work, and she still hasn't. Or maybe you have typed it in but it won't run. Then why not take all the effort out of it and send for the MONITOR DISK. All the main programs in each issue of MONITOR are now available pre-recorded on disk for you. They cost £4.95 which includes postage and packing, send a cheque/postal order made payable to the 'U.K. Atari Computer Owners Club' to Monitor Magazine, P.O. Box 3, Rayleigh, Essex. If you live in Europe add 50p, if outside Europe add £1.00. Please allow 28 days for delivery.

Monitor Disk 8.
Includes: Quickplot, a fast Graphics 8 Plot/Drawto handler. Nightmare Reflections, an exceedingly frustrating adventure. Matchbox, improve your concentration with this memory game. Interrupts, 5 demo programs showing various uses of interrupts.

Monitor Disk 9.
Includes: Keyo, a new typing checker. Multiboot Bootbase, database program for 'Multiboot disks'. Binload, binary loads from Basic. Happytyper, automatic line numbering. Ramdisk, for use with the 130XE. Fast Fill, a speedy shape filling utility.

Monitor Disk 10.
Includes: 3D Maze, escape from the maze in time if you can. PCB Paranoia, destroy your enemies before they get you. Disk Jacket, useful program for making your own disk covers. Chase, an excellent game, not to be missed.

Contact

WANTED issues 1 to 3 of the U.K. Atari Computer Owners Club magazine, any condition. Photocopies or will photocopy and return. Please Help! Shawn McCrindle, 74A Briarhill Road, Prestwick, Ayrshire, KA9 1HY or Phone 0292 74877 after 6.00 pm.

PEN PAL required, write to me and discuss our common interest in all matters Atari. Ray Christie, 15 Heather Grove, Greensborough 3088, Victoria, Australia.

HELP I have 'output' do you have 'input' and live in the Cardiff area, if so and you are interested in joining or forming an Atari User Group, please contact me at Mandeville House, 9 Lewis Street, Canton, Cardiff. You may even find a computer being put to uses that you had not thought of. Look forward to hearing from all you enthusiasts out there in the wilds of Atari land. Raymond Price.

PINBALL fanatics I am willing to lend support if anyone is interested in forming a Pinball User Group, for exchanging games, swopping files, highest scores, etc. Write to Mark Hutchinson, 1 Hollymount, Erinvale, Finaghy, Belfast, BT10 0GL.

BACK ISSUES

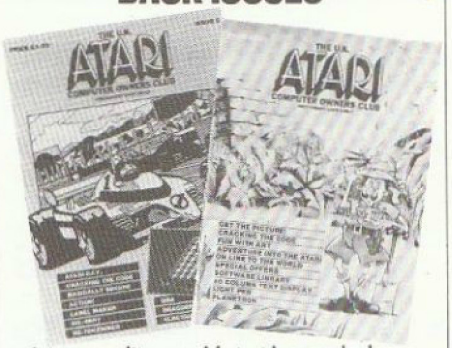
Previous issues of this magazine are obtainable from the club for £1 plus 30p postage each. They contain many interesting and informative articles, hints & tips, program listings for you to input, reviews and practical advice. If you have missed out send for your copies of back issues today! Please note that issues 1,2,3 & 7 are already sold out.

Issue 4.

Includes a complete in-depth look at Display Lists, what they are, how to use them, LMS explained, horizontal and vertical scrolling, etc. Another article shows how to get text on a Graphics 8 screen and gives an example graph to prove the point. A comprehensive review of many of the different types of joystick that are available gives ratings for comfort, action, looks and value. Program listings are a-plenty and include 'Peckman', a BASIC version of a well known arcade game, Stunt Rider in which you must jump your motorbike over the buses, Hex is a two player board game with excellent graphics, and for the more serious minded, you can enjoy designing your own shapes with CAD (computer assisted design).

Issue 5.

The first part of the series on 'Cracking the Code' starts in this issue and covers Binary, Hexadecimal and Decimal mathematics. There is an article on protecting your BASIC programs from prying eyes and an interesting article on hardware modifications to the 800/400 machines to give improved sound and



picture quality, a cold start key and a busy light for your cassette player. Also included is a review of the new programming language 'Action!' showing its potential for creating exciting fast action games. Games listings shown include Gil-bert, which is a 'Q-bert' type game, also Dragonfire in which the player must cross the drawbridge dodging the dragons flaming breath to reach the treasure room. Other listings include a label maker and a QRA locator for Radio Amateurs.

Issue 6.

Includes a useful tutorial showing how to print Micropainter and Versa-writer pictures, also contains a terrific program demonstrating 80 characters across the screen. A new regular column for adventure enthusiasts is started to give reviews of adventure games and give hints and tips on how to play them. Part two of Cracking the Code continues with addressing modes and binary sums. The hardware design for a Light Pen is shown together with some simple programs use with it once you have built it. Fun with Art from Epyx is reviewed and some of the excellent results of using this package are shown. Programs include Planetron and a RTTY listing for use with a short wave band radio, the Atari 850 interface and a signal terminal unit (such as the Maplin TU1000).

Issue 8.

Contains a preview of the new Atari computers. Two new series start, one about how files work and the other 'Starting from Basics' for beginners. Cracking the code continues and concluding part of 'Interrupts' discusses horizontal and vertical scrolling. The adventure column includes reviews of Mask of the Sun and Sorcerer. Other reviews include Conan, Spy vs Spy, Alley Cat and Ghostbusters. Programs are Matchbox, a concentration game, Quickplot, a Graphics 8 Plot/Drawto utility and Nightmare Reflections, an exceedingly frustrating adventure.

Issue 9.

Includes a RAMDISK for the 130XE as well as a review of this excellent machine. Introduction to MIDI, just what is it! KEYO typing checker program. Utility to give binary load files from BASIC. Reviews of TopDOS, Homeword and Mr DO! Overview of FORTH as an alternative to BASIC. Utility to fill in shapes in Graphics 8 and fast too! Profile on Lea Valley Atari Club. HAPPY TYPER gives automatic line numbers and programmable function keys. Utility for indexing 'Multiboot' disks.

ATARI ST

520ST

POWER WITHOUT THE PRICE

THE NEW ATARI 520ST

Under the new leadership of Jack Tramiel (former boss and founder of Commodore Business Machines), Atari Corporation have marked their entry into the world of business/personal computers with a machine which leaves the competition standing. Tramiel's slogan 'Power Without the Price' has been implemented in the manufacture of the new 512K Atari 520ST colour computer which offers the user amazingly high performance at an incredibly low price. Launched as a work-station, this new system incorporates seven software packages as well as the 520ST computer with 512K RAM, mouse controller, high resolution monochrome monitor (640x400), 95 key keyboard (with 18 key numeric keypad), MIDI interface, GEM and a 500K 3½ inch disk drive, all for the package price of only £651.30 (+VAT = £749). Dubbed the 'Mac beater' and the 'Jackintosh' (after Atari's Chief, Jack Tramiel), Atari's new machine has been directly compared with the Apple Macintosh RRP £2595 (+VAT = £2985) which offers similar features and capabilities but at a much higher price. Favourably reviewed by the UK's highly critical specialist computer press, the 520ST is likely to make a great impact in this country as a sophisticated alternative to an IBM PC, APRICOT or APPLE MACINTOSH. Unlike its overpriced competitors, the Atari 520ST can be linked up to a colour monitor to unleash a choice of up to 512 colours. The addition of colour brings out the full potential of graphics packages such as GEM.

USER FRIENDLY GEM OPERATING SYSTEM

The power of the ST is harnessed and made user friendly by the new operating system 'GEM' from Digital Research. GEM stands for Graphics Environment Manager and allows a user friendly colour or B/W graphics interface which closely resembles that of the Macintosh. This similarity extends to the use of moveable resizeable windows, icons to represent objects such as disks and disk drives, and the use of pull down menus and a mouse. The advantage of all this is that the computer becomes extremely easy to use. GEM has now been implemented for the Acorn, ACT, Atari, IBM, ICL, and Olivetti. Software written for GEM on one computer should also run under GEM on another computer. This will enable the market to quickly produce a large library of standard interchangeable software.

FREE SOFTWARE AND FUTURE EXPANSION

The Atari 520ST comes supplied with seven free software packages as listed below: 1) TOS -Tramiel Operating System based on CPM 68K. 2) GEM Graphics Environment Manager by Digital Research (DR) giving a WIMP (Window, Icon, Mouse, Pull down menu) environment. 3) DR GEM Paint for creating graphics masterpieces. 4) DR GEM Write for word processing. 5) Logo learning language to enable you to write your own programs easily using turtle graphics. 6) DR Personal Basic a powerful user friendly version of the Basic programming language. 7) BOS operating system giving you access to dozens of business applications packages already available on the market. Designed with future expansion in mind, the ST also features a host of different interfaces to the outside world and an impressive list of accessories is planned. Atari will soon be releasing a 1000K (1MB) 3½ inch disk drive, and a 15MB hard disk storage system as well as a mass storage compact disk (CD) player capable of storing an entire 20 volume encyclopedia on one disk. A full range of inexpensive printers are planned including dot matrix, daisywheel and thermal colour printers. With its unbeatable graphics, speed and software at a price which is far below that of any comparable personal computer currently on the market, the ST is all set to do battle with the competition. To receive further details of the ST from Silica Shop, just fill in the coupon below with your name and address details and post it to us.

Silica Shop Price: £651.30 + £97.70 VAT = £749.00 This price includes:

- ★ 512K RAM
- ★ B/W MONITOR
- ★ MOUSE
- ★ 500K 3.5" DISK DRIVE
- ★ GEM
- ★ KEYBOARD (95 KEYS)

£749

ATARI 520ST SPECIFICATION

MEMORY
512K RAM (524,288 bytes)
16K ROM expandable to 320K
Port for addnl 128K plug-in ROM cartridges
200K TOS operating system

GRAPHICS
Individually addressable 32K bit-mapped screen with 3 screen graphics modes:
320x200 pixels in 16 colours (low resolution)
640x200 pixels in 4 colours (med resolution)
640x400 pixels in monochrome (high res)
16 shades of grey in low res mode
512 colours available in low/medium res
8 levels of each in red, green and blue

ARCHITECTURE
4 custom designed chips:
GLUE Chip - MMU Memory Mngmnt Unit
DMA Controller - Graphics Processing Unit
16/32 bit Motorola 68000 processor at 8MHz
eight 32 bit data registers
eight 32 bit address registers
16 bit data bus/24 bit address bus
7 levels of interrupts/56 instructions
14 addressing modes/5 data types

DATA STORAGE
High speed hard disk interface
Direct memory access 1.33 Mbytes per second
CD (Compact Disc) interface
Built in cartridge access
Dedicated floppy disk controller

DISK DRIVE
500K (unformatted) s/sided 3½" floppy d/drive
349K (formatted) storage capacity

SOUND AND MUSIC
Sound Generator
Frequency control from 30Hz to above audible
3 voices (channels) in wave shaping sound in addition to a noise generator
Separate frequency and volume controls
Dynamic envelope controls
ADSR (Attack, Decay, Sustain, Release)
Noise generator
MIDI interface for external music synthesizers

KEYBOARD
Separate keyboard microprocessor
Standard QWERTY typewriter styling
Ergonomic angle and height
95 keys including 10 function keys
Numeric keypad — 18 keys including ENTER
One touch cursor control keypad

MONITOR
12" screen - high res monochrome monitor
640x400 monochrome resolution

VIDEO PORTS
Display - Low Resolution - 40 columns
Med/High Res - 40/80 plus cols
Medium res RGB (Red/Green/Blue) output
High resolution monochrome (Black & White)

COMMUNICATIONS
Bidirectional centronics parallel interface for printers, or modems capable of input/output
RS232C serial modem/printer interface
VT52 Terminal Emulation Software
Maximum Baud Rate up to 19,200
High speed hard disk interface
Floppy disk controller (Western Digital)
2 joystick ports (one for 2 button mouse)
MIDI interface for external music synthesizers

GEM WIMP ENVIRONMENT
WIMP - Window Icon Mouse Pop-down menus
Two button mouse controller
Icons/Pull down menus/Windows
GEM VDI - Virtual Device Interface
GEM AES - Application Environment Services
GEM BBT - Bit Block Transfer
Real time clock & calendar

SOFTWARE
GEM environment
with user friendly Macintosh style operation
'TOS' - Tramiel Operating System
Atari's own system based on CP/M 68K with hierarchical directory & file structure plus a host of MS DOS & UNIX command structures
'BOS' - Business Operating System
to run any standard BOS business programs
GEM desktop
with GEM PAINT graphics mgmt system and GEM WRITE word processor
Personal BASIC and DR Logo
originally written by Digital Research (DR)
Very much like those on other machines except for the extensive use of pull down menus, mouse control and windows

VARIOUS
Dimensions: 470mmx240mmx65mm
Replaceable external power supplies
Expansion: 3½" floppy disk drives 500K/1.000K (two drives can be connected)
3½" 15Mb hard disk
CD (compact laser disc)
Dot matrix & d/wheel printrs (black)
Thermal dot matrix (colour)
RGB & monochrome monitors

LANGUAGES
BASIC & LOGO supplied
Many others will soon be available, including:
Assembler, BCPL, C, Cobol, Compiled Basic
Lisp, Modular-2 and Pascal

Note: Some of the above specifications are pre-release and may therefore be subject to change

MACINTOSH v F1e v 520ST

"Imagine a Fat Mac - the 512K Apple Macintosh - but with a bigger screen, a far bigger keyboard with numeric keypad, cursor and function keys, and colour. That gives you some idea of what the Atari 520ST is like, except for two important things. First the Atari seems faster. Second the Atari system is about one third of the price."
June 1985 - Jack Schofield - **PRACTICAL COMPUTING**

FEATURES OF BASIC SYSTEM	APPLE MACINTOSH	APRICOT F1e	ATARI 520ST
Price includes B/W Monitor	YES	NO - extra £200	YES
Keyboard size mm (L x D x H)	330 x 147 x 50	450 x 167 x 28	470 x 240 x 80
Keyboard size ins (L x D x H)	13 x 5¾ x 2	17½ x 6½ x 1	18½ x 9½ x 2½
3½" D/Drive (Unformatted)	500K	500K	500K
3½" D/Drive (Formatted)	399K	315K	349K
WIMP (Window, Icon, Mouse ..)	Apple	ACT - Activity	GEM
Real-time Clock	YES	YES	YES
Polyphonic Sound Generator	YES	NO	YES
RS232 Serial Port	YES	YES	YES
Centronics Parallel Printer Port	NO	YES	YES
Dedicated Floppy Disk Controller	NO	YES	YES
Hard Disk DMA Interface	NO	YES	YES
Full stroke keyboard	YES	YES	YES
Number of keys on keyboard	59	92	95
Numeric Keypad	NO	YES (16 Keys)	YES (18 keys)
Cursor Control Keypad	NO	YES	YES
Function keys	NO	10	10
16-bit processor	68000	Intel 8086	68000
Processor running speed	8MHz	4.77MHz	8MHz
RAM size	512K	256K	512K
Number of graphics modes	1	4	3
Number of colours	Monochrome	16	512
Max Screen Resolution (pixels)	512 x 342	640 x 256	640 x 400
Mouse included	Single Button	NO - extra £95	Two Button
Replaceable External Power Pack	NO	NO	YES
Cartridge Socket	NO	NO	YES
Joystick Ports	NO	NO	YES (two)
MIDI Synthesiser Interface	NO	NO	YES
Monitor Size	9"	9" - extra £200	12"
RGB Video Output	NO	YES	YES

System Cost with: Mouse - Monochrome Monitor - 512K RAM - 500K Disk Drive

Price of basic system (exc VAT)	£2595+VAT	£595+VAT	£652+VAT
+ Mouse	Included	£95+VAT	Included
+ Monochrome Monitor	Included	£200+VAT	Included
+ Expansion to 512K RAM	Included	£295+VAT	Included
Price of complete system (exc VAT)	£2595+VAT	£1185+VAT	£652+VAT
PRICE rounded down including VAT	**£2,984**	**£1,362**	**£749**

PRESS COMMENT

"Atari's new corporate image as an aggressive low cost computer maker is likely to mirror that of Commodore where Mr. Tramiel established the maxim that 'Business is war'."
August 21st 1984 **FINANCIAL TIMES**

"This is the only personal computer I know of that comes with a Midi interface as standard."
Peter Bright March 1985 **PERSONAL COMPUTER WORLD**

"The (GEM) version running on the Atari 68000 machines will have the additional advantage of leaving the PC version standing."
April 6th 1985 **PERSONAL COMPUTER NEWS**

"It would seem that GEM offers the ideal operating system."
March 7th 1985 **POPULAR COMPUTING WEEKLY**

"I found it (GEM) extremely easy to use and was very impressed with the way in which it disguises the unfriendly hardware and operating systems lurking under the surface."
Peter Bright Feb 1985 **PERSONAL COMPUTER WORLD**

"The electronics in the machine are a work of art ... The heart of the 520ST is a Motorola 68000, one of the most powerful 16-bit processors around and in many respects it is close to being a 32-bit chip ... when the machine appears in the shops, I'll be at the front and of the queue to buy one."
Peter Bright June 1985 **PERSONAL COMPUTER WORLD**

"This machine is significantly more powerful than an IBM PC ... If it's possible to design a sure-fire winning machine, this is it."
May 11th 1985 **PERSONAL COMPUTER NEWS**

"... the use of GEM makes the new range of Atari computers so similar to the Macintosh (with the added attraction of colour), that they are already being called 'Jackintoshes'."
May 2nd 1985 **COMPUTING**

"The new Atari ST computers truly represent to the consumer what Jack Tramiel is saying — easy-to-use computing power without the price."
March 1985 **ANALOG COMPUTING**

"It (the ST) uses the most modern technology that is affordable, in a package that gives a professional impression."
May 23rd 1985 **POPULAR COMPUTING WEEKLY**

"The Atari ST is one of the most elegant designs I have seen ... Atari has used an original and elegant method of memory management which should make the ST faster than any other PC on the market - in any price bracket ... The 64K dollar question is would I go out and spend money for one? To which the only answer is Try and stop me!"
John Lambert July 1985 **ELECTRONICS & COMPUTING**

"The 520ST is technically excellent ... The 520ST hardware is the new standard by which others will be judged."
July 1985 **YOUR COMPUTER**

SILICA SHOP LTD, 1-4 The Mews, Hatherley Road, Sidcup, Kent, DA14 4DX

SEND FOR FREE ATARI ST LITERATURE

SILICA SHOP

WE ARE THE UK's No1 ATARI SPECIALISTS

At Silica we have been successfully dedicated to Atari ever since their products first appeared on the UK market. We can attribute our success largely to the Atari specialisation which we practice and to the user back-up we provide. Rest assured that when you buy a piece of Atari hardware at Silica you will be fully supported. Our mailings giving news of software releases and developments will keep you up to date with the Atari market and our technical support team and sales staff are at the end of the telephone line to deal with your problems and supply your every need. With our specialist bias, we aim to keep stocks of all the available Atari hardware, software, peripherals and accessories. We also stock a wide range of Atari dedicated books and through us, the owners on our list can subscribe to several American Atari dedicated magazines. We can provide a full service to all Atari owners and are now firmly established as the UK's NUMBER ONE Atari specialists. Here are just some of the things we can offer to our customers:

- ★ FREE POST & PACKING ON MAIL ORDERS
- ★ FREE NEXT DAY SECURICOR DELIVERY
- ★ INFORMATION MAILING SERVICE
- ★ TECHNICAL SUPPORT TEAM
- ★ HIGHLY COMPETITIVE PRICES
- ★ AFTER SALES SUPPORT SERVICE
- ★ REPAIR SERVICE ON ATARI PRODUCTS

If you would like to be registered on our mailing list as an Atari computer owner, or as a person interested in buying an Atari machine, let us know. We will be pleased to keep you up to date with new Atari developments free of charge. So, return the coupon today and begin experiencing a specialist Atari service that is second to none.

SILICA HOTLINE ☎ **01-309 1111**

To: Silica Shop Ltd, Dept ATCOC 1285, 1-4 The Mews, Hatherley Road, Sidcup, Kent, DA14 4DX

PLEASE SEND ME FREE LITERATURE

ON THE NEW ATARI 520ST COMPUTER

Mr/Mrs/Ms: ..... Initials: ..... Surname: .....

Address: .....

.....

.....

Postcode: .....

Do you already own a computer
If so, which one do you own? .....