TYNE & WEAR

ATARI 8 BIT

USER GROUP

Issue 23

September/October 1996

Reminder

TWAUG NEWSLETTER is published bi-monthly, around mid-month of (Jan, Mar, May, July, Sept and Nov.)

It is printed and published by TWAUG, no other publishing company is involved.

Opinion expressed by authors, in this newsletter, is their own opinion and do not represent the views of TWAUG.

Contacts:

Alan Turnbull on: 01670 - 822492

or Max on: 0191 - 586 6795

New Postal address:

TWAUG

c/o J.Matthewson

80 George Road

Wallsend, Tyne & Wear,

NE28 6BU

TWAUGs APOLOGY

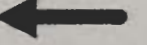
We believe all our subscribers received issue 22 very late, we therefore apologize for that delay. It was partly due to me (Max), only for about three to four days, I mentioned it in one of the replies to a letter that I wasn't feeling to good at the time. But the delay of nearly two weeks was due to the postal dispute, we hope you have received issue 22 now.

Change of Postal Address

I have no doubt, you all received in issue 22 a leaflet, informing you that we intend to discontinue with the Post Box address.

We had a number of phone calls from subscribers telling us that they never received the orders, which they had addressed to the PO BOX, unfortunately we never received them. As mentioned in the leaflet in issue 22, a lot of letters and parcels have gone astray and when we inquired about the lost mail, all we got was we will look into it, but nothing was ever heard of it since.

This is the reason we want you to post your correspondence direct to John, and hope it is the end of the lost mail.

 John's address

PUBLISHING!

This new look newsletter is set up with the Desktop Publishing program "TIMEWORKS 2", on the Mega 1 ST with 4 meg memory. Files are converted to ASCII and transferred to the ST with TARI-TALK. Those files are then imported into the DTP and printed with the Star/ LC24-100 at 360dpi, with excellent result.

TWAUG

NEEDS  YOU

TWAUG subscriptions

Home	1 Copy	£2.50
- DO -	6 Copies	£12.50
Europe	1 Copy	£2.50
- DO -	6 Copies	£13.50
Overseas	1 Copy	£3.50
-- DO --	6 Copies	£16.00

Issue 24 is due mid-November

ISSUE CONTENT

DON'T LET BASIC BUG YOU

Part VI of MIKE BIBBY's guide through the micro jungle

In PART V we looked at how to create loops using a conditional statement and a GOTO. However, if you just want your micro to do something a fixed number of times, there is another technique you can use, the FOR ... NEXT loop.

If you have a number of lines of a program that you want repeating for a definite number of times you mark them out by putting the FOR statement at the beginning and the NEXT statement at the end of those lines.

FOR my

When the micro reaches a FOR it knows it has a loop on its hands. It will repeat the lines (or code, as the professionals say) between the FOR and the NEXT as many times as needed. To do this, the micro needs to use a variable as a counter to keep track of how often the loop has been performed.

In our previous loops we've always used a numeric variable for our counter - *number*. Each time the loop was performed we increased *number* by one until we reached our finishing condition.

In a FOR ... NEXT loop the variable you use for your counter increases automatically on each repetition of the loop. However, you need to tell the micro where to start and where to finish. To see how we do this in practice, let's look at Program I, which prints out HELLO 10 times.

And now

Lines 30 and 50 mark out the lines we want repeating (line 40). Line 30 reads:

30 FOR NUMBER = 1 TO 10

The FOR indicates the beginning of the loop. This is followed directly by the counter variable, in this case *number*. After the '=' sign the 1 to 10 tells the micro to start *number* at 1 and keep on increasing it by one each time the loop is repeated until it gets

NEXT trick...

DON'T LET BASIC BUG YOU continued

```
10 REM PROGRAM I
20 PRINT CHR$(125)
30 FOR NUMBER=1 TO 10
40 PRINT "HELLO"
50 NEXT NUMBER
60 PRINT "GOODBYE"
```

Program I

past 10.

The loop is then finished and the micro carries on with the rest of the program, in this case line 60. The outcome of all this is that HELLO is printed 10 times followed by a final GOODBYE.

The micro's thought processes go like this:

```
NUMBER = 1 PRINT "HELLO"
Increase NUMBER
NUMBER = 2 PRINT "HELLO"
Increase NUMBER
NUMBER = 3 PRINT "HELLO"
```

and so on until:

```
NUMBER = 9 PRINT "HELLO"
Increase NUMBER
NUMBER = 10 PRINT "HELLO"
Increase NUMBER
NUMBER = 11 But the loop is to 10
so go on to line 60
```

Let's learn some jargon:

- *What we've called the counter variable is, not surprisingly, called the loop variable.*
- *The "limits" of the loop - in this case 1 to 10 - are called the loop parameters.*
- *The lines of the code to be repeated are termed the body of the loop.*
- *When you finish a loop and continued with the rest of the program we say that you have dropped out of the bottom of the loop.*

Notice that we've put the loop variable, number, after NEXT in line 60. Some Basics let you leave this out - not so the Atari.

All the above has been a rather long-winded explanation of a simple method of getting the computer to do something a fixed number of times. Try the following versions of line 30., and keep a careful count of the number of HELLOs you optain. Are they what you expected?

```
30 FOR NUMBER = 1 TO 20
30 FOR NUMBER = 10 TO 20
30 FOR NUMBER = 0 TO 20
30 FOR NUMBER = 11 TO 20
30 FOR NUMBER = 0 TO 11
```

DON'T LET BASIC BUG YOU continued

As will be obvious from the above, the loop variable doesn't have to start at 1. Just to warn you of a possible source of future errors, try changing line 30 to:

```
30 FOR COUNTER = 0 TO 11
```

Assuming that you haven't changed line 50 from the original Program I, you'll get an error message. This is because the loop variable you've specified in the FOR statement (counter) doesn't match the one after the NEXT (number).

Now try Program II.

```
10 REM PROGRAM II
20 PRINT CHR$(125)
30 FOR LOOP=1 TO 10
40 PRINT LOOP
50 NEXT LOOP
```

Program II

If you recall, the loop parameter increases by one each time the loop is repeated. In a burst of wild originality I've called the loop parameter loop. The first time through the loop, loop is 1, so line 40 prints out the value 1. Then loop is increased to 2 since it is the counter, so line 40 prints out 2, and so on.

Once you've worked out what is happening here try adding:

```
60 PRINT LOOP
```

The new line prints out the value of loop after the loop has ended. Can you explain the result?

```
10 REM PROGRAM III
20 PRINT CHR$(125)
30 FOR LOOP=1 TO 10
40 PRINTLOOP,LOOP*LOOP,LOO
P*LOOP*LOOP
50 NEXT LOOP
```

Program III

Program III prints out the squares and cubes of the numbers up to 10. Rather nice isn't it? Of course, there's no need for you to stop at 10 - try increasing it to 100. That's the good thing about the loops - you can get the micro to do a considerable amount with very little coding on your part. Program IV will print out whatever multiplication table you want.

```
10 REM PROGRAM IV
20 PRINT CHR$(125)
30 PRINT
40 PRINT "Which table do you want ?"
50 INPUT NUMBER
60 FOR LOOP=1 TO 12
70 PRINT LOOP;" multiplied by "; NUM
BER;" is ";LOOP*NUMBER
80 NEXT LOOP
```

Program IV

DON'T LET BASIC BUG YOU continued

Can you alter line 60 of Program III so that the output starts with a 10 and decreases to 1?

Now try running Program V.

```
10 REM PROGRAM V
20 PRINT CHR$(125)
30 PRINT CHR$(65)
40 PRINT CHR$(66)
50 PRINT CHR$(67)
```

Program V

Even if you don't fully understand what's going on, I bet you can still guess what

PRINT CHR$(68)

would give you!

CHR$ stands for "Character String" though I always read it as "Chris", so I would pronounce:

PRINT CHR$(85)

as "print Chris eighty-five". The code number can be stored in a variable, so:

NUMBER = 65

PRINT CHR$(NUMBER)

will work. You see, every character you can put on the screen has its own code number. The code for A is 65, for B is 66 and so on. CHR$( ) takes the code and turns it into a character string - that is, a string a single character long.

These numbers have been standardised in a table called, rather grandly, the American Standard Code for Information Interchange. It's known as Ascii - pronounced "Ascey" - for short. If, however, like me you can never make head nor tail of tables of information, you'll be glad to know that you can use a Basic word called ASC( ) to tell you the number, or Ascii code, of the character you're interested in.

You just put the letter you want inside the brackets - in quotes of course, as we always do with strings. For example, we ask the micro to print out the code for A with:

PRINT ASC("A")

which, if you remember to press Return (and I'm not going to remind you from now on!) will give you 65, the code for A. ASC stands for Ascii, so read the example above as "PRINT Askey A".

```
10 REM PROGRAM VI
20 PRINT CHR$(125)
30 DIM STRING$(1)
40 PRINT "A letter ?"
50 INPUT STRING$
60 PRINT "ASCII code for ";STRING
$;" is ";ASC(STRING$)
70 GOTO 40
```

Program VI

DON'T LET BASIC BUG YOU continued

Program VI generates the Ascii code for the character you input. Try inputting a string of more than one character and see what happens. Program VII shows the printable Ascii codes between 32 and 122. There are others, but for the moment we'll ignore them. Notice the loop parameters in line 30.

```
10 REM PROGRAM VII
20 PRINT CHR$(125)
30 FOR LOOP=32 TO 122
40 PRINT CHR$(LOOP);
50 NEXT LOOP
```

Program VII

Remember, you don't have to start a FOR ... NEXT loop with the value 1. However, it's sometimes easier to visualise what's going on if the loop does start with 1, or perhaps zero. For instance, Program VIII prints out the whole alphabet in capitals!

```
10 REM PROGRAM VIII
20 PRINT CHR$(125)
30 FOR LOOP=-65 TO 90
40 PRINT CHR$(LOOP)
50 NEXT LOOP
```

Program VIII

However, I prefer Program IX, which performs the same task.

```
10 REM PROGRAM IX
20 PRINT CHR$(125)
30 OFFSET=64
40 FOR LOOP=1 TO 26
50 PRINT CHR$(OFFSET+LOOP);
60 NEXT LOOP
```

Program IX

What happens is that, since offset is 64 throughout the loop, line 50 prints out the CHR$ of loop plus 64.

For example,

for loop = 1, CHR$(65) is printed;

for loop = 2, CHR$(66) is printed

and so on.

I admit there's a bit of mathematical jiggery-pokery involved, but when I'm dealing with the alphabet the numbers 1 to 26 mean for more to me than 65 to 90.

Granted there's one more line than in Program VIII, but it is far easier to alter the program if, say, I happen to get my figures wrong. To demonstrate this, change line 30 to:

30 OFFSET= 96

Hey presto, lower case! The codes for the lower case alphabet lie from 97 to 122. Try altering Program VIII to print out in lower case, and you'll

DON'T LET BASIC BUG YOU continued

see it involves much more work.

Of course you could have had *offset*=65 and *loop* from 0 to 25, but that doesn't mean as much to me - I always think of the alphabet in terms of 26! While we're on the subject of offsets, let's have a look at Program X. This prints the numbers from 10 down to 1 rather than from 1 to 10.

```
10 REM PROGRAM X
20 PRINT CHR$(125)
30 FOR LOOP=1 TO 10
40 PRINT 11-LOOP
50 NEXT LOOP
```

Program X

What happens is that instead of just printing out the loop variable line 30 subtracts it from 11 first. So:

when loop=1, 10 is printed (11-1)
when loop=2, 9 is printed (11-2)

and so on until:

when loop=10, 1 is printed (11-10).

Here we are using 11 as a sort of offset.

Try using this idea of taking the loop variable from a number to alter Program VIII and IX to print the alphabet in reverse, Z to A. Before we leave Program X, I must make the point that I would normally write line 40 as:

40 PRINT(11 - LOOP)

The brackets do not affect the outcome. They're used here simply as a "container" for the mathematics. I prefer this tidier approach, even if it's not strictly necessary.

Sometimes, however, the use of brackets is vital. For instance:

PRINT (8-2)*3

and

PRINT 8-(2*3)

give totally different results. What happens is that the micro performs the sums inside the brackets first, then does the rest.

So in the first example the micro says to itself 8 minus 2 is 6, multiplied by 3 gives 18, whereas in the second it says 2 multiplied by 3 is 6, subtracted from 8 leaves 2. So my amended line 40 tells the computer to do the sum first, then print the answer. As I've said, in this case it's not strictly necessary, but such good habits may prevent you inadvertently dropping into error later.

```
10 REM PROGRAM XI
20 PRINT CHR$(125)
30 FOR LOOP=0 TO 10 STEP 2
40 PRINT LOOP
50 NEXT LOOP
```

Program XI

DON'T LET BASIC BUG YOU continued

Have a look at Program XI. This prints out the numbers 0, 2, 4, 6, 8, 10. That is, we go from 0 to 10 in steps of 2. Line 30 holds the secret. You see, we've assumed that in FOR ... NEXT loops the loop variable - we've always used loop-increases, or steps up by one, each time through the loop.

Actually we can tell the computer how much is added each time by tagging STEP onto the end of our previous FOR line. In line 30 we have specified a STEP of 2, so 2 is added to the value of the loop variable each time. Change line 30 to:

30 FOR LOOP=1 TO 10 STEP 2

and you get 1, 3, 5, 7 and 9 printed out.

Notice that 10 is never printed - this is because when loop is 9 and you come to NEXT loop, you increase it by 2, obtaining 11. This is outside the loop parameters, so you drop through the bottom of the loop - that is, the loop ends. You can actually use the idea of STEP to decrease the loop variable - you just use a negative STEP.

Program XII uses this technique to print out the numbers 10 down to 1, far more simply than in Program XI.

```
10 PROGRAM XII
20 PRINT CHR$(125)
30 FOR LOOP=10 TO 1 STEP-1
40 PRINT LOOP
50 NEXT LOOP
```

Program XII

Notice that the loop parameters now go from 10 to 1. The larger number comes first, since we are decreasing the parameters each time. Adding -1 is equivalent to taking 1 away. You don't even have to increase STEP by whole numbers. To prove it, try changing line 30 of Program XII to:

30 FOR LOOP=1 TO 10 STEP 0.5

Can you see what's happening?

- *Now that we've covered the fundamentals of loops we'll continue next time by using them in a variety of ways.*

Atari Portfolio Club

Are you a Portfolio user? If so why not join the new Portfolio user Club, membership is free.

For more information contact:

Paul Finch, 16 Cedars RD., Morden, Surrey, SM4 5AB.

Enclosing a SAE for membership entry form and details.

GRAPHICS - DISPLAY LIST

Just before I completed the last issue of the newsletter, I received a phone call from a subscriber who wanted to know if I could include an article about Graphics Display List. I agreed and said it would be a very good idea. I am sure that there are a lot of computer users who have no idea how to set up a graphics screen. At the time I was also thinking ahead to the next issue, I was at a loss at the time as I had no new material on hand for this issue.

After the chap rang off, I sat thinking about this article I was to write, when it hit me. This really is a mammoth task, far too big for a chap like me to take on. I am familiar with graphics display list, but to write about it is really beyond me, only experts are good enough to do that. Whilst pondering about it, it came to me, why not consult the experts who taught me all I know in the first place, "the magazine". Yes, the Old Atari User magazine, a lot of 8 bit users will know about it but I am also sure that a lot of people never heard about this mag.

This article, about the Graphics capability of the Atari 8-bit, was first published in July 1985 and the author was MIKE ROWE.

We are still looking for contibutions from our members who would like to supply us with some material, anything new or old is welcome.

Antic ...the reason Atari graphics pack such a mighty punch

GRAPHICS - DISPLAY LIST

One of the Atari's most reknowned and spectacular features is its graphics capability. The machine has 16 different graphics modes and can display up to 16 colours from Basic (256 using machine code).

This is more than any of its rivals and more than many computers costing thousands of pounds. The reason the Atari is able to perform these feats is the inclusion of a chip called Antic to look after screen display.

This is a microprocessor in its own right and runs alongside the 6502 main microprocessor, freeing that for the user program. In addition there is the GTIA chip, which is also a microprocessor. This creates the famous Atari player-missile graphics and interfaces the computer to the TV display.

For those of you who are new to your Atari the 16 modes consist of

Basic mode number	Antic mode number	Text or graphics	Number of colours	Screen Columns	Rows full	Rows split	Bytes of memory needed
0	2	TEXT	2	40	24	-	993
1	6	TEXT	5	20	24	20	513
2	7	TEXT	5	20	12	10	261
3	8	GRAPHICS	4	40	24	20	273
4	9	GRAPHICS	2	80	48	40	537
5	10	GRAPHICS	4	80	48	40	1017
6	11	GRAPHICS	2	160	96	80	2025
7	13	GRAPHICS	4	160	96	80	3945
8	15	GRAPHICS	2	320	192	160	7900
9	15	GRAPHICS	1*	80	192	-	7900
10	15	GRAPHICS	8	80	192	-	7900
11	15	GRAPHICS	16	80	192	-	7900
12	4	TEXT/GR	5	40	24	20	1152
13	5	TEXT/GR	5	40	12	10	664
14	12	GRAPHICS	2	160	192	160	4296
15	14	GRAPHICS	4	160	192	160	8138

*=16 Shades of 1 colour

Note that graphics modes 12-15 are available directly from Basic only on the XLs. They can only be obtained on the 400/800 computers by creating the mode yourself.

Figure I: Graphics modes

GRAPHICS - DISPLAY LIST

five modes that display text and 11 modes that display graphics. These are shown in Figure I.

You may have noticed that there are two kinds of mode number, Basic and Antic. The Basic number is that used in a graphics call from a Basic program. For example Graphics 0 gives you the standard 40x24 text mode.

The Antic mode number is the one stored in memory to be used by the Antic chip to tell it what kind of screen to display. This is calculated from the Basic mode number and stored in the correct location in memory by the computer's operating system -the Antic number of Basic graphics Mode 0 is in fact 2. Using the Antic mode numbers directly without a Basic graphics call will be explained in later articles.

Don't ask me why Atari had to make the two numbers different, but they did and we're stuck with it. From now on, when I refer to graphics modes I mean the Basic mode and if I want to refer to the Antic mode I will specify Antic.

How does the Antic chip work? A television picture is created by a beam of electrons hitting a fluorescent screen on the inside of your TV tube (oversimplified). The beam is made to scan horizontally in sequential lines across the screen and the whole screen is covered 50 times a second.

A normal TV picture consists of 625 of the lines (in fact it consists of 312 interlacing, alternating lines). The computer display, to avoid overscanning the TV and losing data, consists of only 192 lines, leaving a gap at the top and bottom of the screen.

Antic is able to control each scan line individually and up to 320 individual pixels horizontally. A pixel is a single point on the screen created by the computer and therefore the smallest dot it can make.

In between each horizontal scan of a line there is a small delay - the horizontal blank. Also between each time the screen is drawn there is another delay - the vertical blank. More of these later.

The higher resolution modes (192 vertical resolution, say Graphics 8) use one scan line per horizontal row of the screen. However, other modes use up to 16 scan lines per line of the graphics mode.

The scan lines used are:

GRAPHICS - DISPLAY LIST

Basic mode	Vertical resolution	Scan lines/ mode line
0	24	8
1	24	8
2	12	16
3	24	8
4	48	4
5	48	4
6	96	2
7	96	2
8-11	192	1
12	24	8
13	12	16
14	192	1
15	192	1

The next question is, how does Antic know what to display? The answer lies in the display list, a small machine code program interpreted by Antic to give the display. It tells the chip two main things:

- The Antic graphics mode number for each line.
- The memory location of the screen display.

It is normally created and manipulated by the computer's operating system and the Basic programmer can forget it.

The whereabouts of the display list is stored in rather a complicated way, in memory locations decimal 560 and 561, because a computer does not work in decimal (base 10) as we do. It works in binary numbers (base 2). These are often expressed as hexadecimal (base 16).

Every memory location can store a number between 0 and 255. Therefore to express numbers greater than 255 you must use two memory locations. So to store a number such as 42000 you must split it into two parts. This is done by firstly finding the number of times 256 will divide into it and secondly the remainder.

The first number is known as the high byte of the number and the remainder is the low byte. They are stored in memory in the order low byte, high byte. For example you get 42000/256=164 remainder 16. The high byte is 164 and the low byte 16.

If 42000 was the location of the display list then 560 would contain 16 and 561 would contain 164 (if there is no remainder then 0 must be stored in 560).

Conversely, to find where the display is located you multiply the number in location 561 by 256 and add this to the number in location 560, that is PEEK(561)*256+PEEK(560) gives the location of the display list.

Most display lists are very short, usually less than 100 bytes. The

GRAPHICS - DISPLAY LIST

Decimal	Hex	
112	70	) 3 lines
112	70	) each of 8 blank
112	70	) scan lines
66	42	=64 (LMS instruction) +2 (Graphics 0 line)
64	40	) Screen memory location
156	9C	) =64+156*256
2	02	) 23 lines the same
"	""	) i.e.23 Basic Graphics 0 lines
65	41	=64+1 End of display list & JuMP to
32	20	) Memory location of start of list
156	9C	)=32+156*256

Figure II: Graphics 0 display list

display list used for Graphics 0 is typical and is shown in Figure II.

To some extent the display list is fairly self-explanatory, however, a few things need expanding. Firstly, the LMS instruction. This means Load Memory Scan and tells Antic to look at the next two instructions to find where in memory the screen should be displayed from.

The above display list has only one LMS instruction but a display list can have several of these pointing to different memory locations, and can even have a different LMS for each mode line.

Therefore any mode number can be added to an LMS instruction to tell Antic to look for its display data wherever you wish. The above display list starts with three lines, each of eight blank scan lines to give 24 blank scan lines at the start of the list.

All the standard graphics modes start with this. The number 112 ($70) is only one of several "blank line" instructions:

The end of the display list can be split into three numbers starting with a 65 ($41). This can be divided into 1+64. The 1 tells the display list to jump and the 64 is an LMS telling Antic that a memory location follows. The next two numbers are therefore the memory location that the list

GRAPHICS - DISPLAY LIST

jumps to, in this case the start of the list. These two numbers will be the same as in memory location 560 and 561 respectively, as they point back to the beginning of the display list.

Decimal	Hex	Number of blank scan lines
112	70	8
96	60	7
80	50	6
64	40	5
48	30	4
32	20	3
16	10	2
0	00	1

Other instructions may also be included in the list and the following table gives the instruction codes that can be included in a display list by adding it to the Antic mode number. We'll see more of these in later articles.

Decimal	Hex	Instruction
16	10	Horizontal scroll
32	20	Vertical scroll
64	40	LMS
128	80	Jump to the display list interrupt

This is all very interesting, I hear you say, but of what use is it and do I really need to know all this? Well, if you are happy to have just the 16 simple modes provided then no!

However, much more spectacular and attractive displays become available if you can understand this and know how to alter things to your heart's desires. This is done by creating your own custom display list and mixing modes on the same screen and by creating things called display list interrupts.

More about these next time.

I received this poem from my friend in New Zealand. This poem was discovered in the December 1992 issue of Presstime, an American newspaper magazine. It was strictly anonymous.

I have a spelling checker,

It came with my PC:

It plainly marks four my revue,

Mistakes I cannot sea.

I've run this pome threw it,

I'm sure your pleas too no.

It's letter perfect in its weigh,

My checker tolled me sew.

REVIEWS

by KEVIN COOKE

More game reviews for your perusal! Sorry for the shorter-than-usual column this time but college pressure has, yet again, left me with very little time. Hopefully it will be a little meatier next time! Until then, give the following three reviews an eyeball!

Title:

GRANDMA'S HOUSE Sold by:

Micro Discount,

265 Chester Road, Streetly,

West Midlands B74 3EA,

ENGLAND.

Tel: 0121 - 353 5730

Price: 6 Pounds 95p (+ P&P)

Although rare nowadays, Grandma's house is a practically new "educational" game for kids aged 4-8. I say "practically new" because, as far as I know, Grandma's House has only been available in extremely limited supplies up to now. Hopefully Derek Fern's stocks are a little more bulky than this!

Grandma's House, released by Spinnaker (of the good ol' US of A) some time ago is a basic concept but a fun one at that. The game starts off with a lively nursery-style tune - a press of the SPACE bar exits the credits and loads the game proper. The first screen to appear is one containing around 30 different people, ranging in age from very young babies to the elderly. By moving a selector box around the screen, the player can select two players for them to later control. As each one is selected, they appear in one of two doorways at the bottom of the screen. When two characters have been selected, the screen turns blue and the disk is accessed some more. At this point, a wireframe 3D empty house appears. It is completely unfurnished and the only solid-looking items in it are your previously two selected characters and good old Grandma!

The object of the "game" is to visit one of 8 different locations and pick up items with which to furnish the house. This is done by simply moving your character over the wanted object, pressing the fire button to pick up the object and then

REVIEWS

taking it back to the house. Once in the house, the object can be dropped practically anywhere!

Although only one of the two selected characters can be controlled at once, the player can take control of the other by a simple keypress. This could allow two children to take it in turns, or one child to use either character. The eight visitable locations are all fairly well drawn, although only in 4 colours. Sometimes, as your character also has to share these colours, he/she can be hard for children to identify. One of the eight visitable locations is the original character selection scheme so the player can even pick some of these characters to put into Grandma's house.

Sound throughout the game is excellent with each scene having good music tunes which children will enjoy.

There's very little to say about this program except that it's very good and your children are bound to enjoy it. Even my three-year old sister managed to play it - she probably enjoys it as much as MY FIRST ALPHABET! Although there is a short disk access between each change of scene, this doesn't seem to prevent children from enjoying this program at all. Who says children aren't patient?!!

The only minor niggle is that some of the items in the visitable locations are too big for the character to pick-up. This can annoy the child a little, especially if they really wanted the lion or the car on the top floor of the house!

Apart from this, I can't recommend this educational "game" enough if you have children within the recommended ages. Just make sure that after you've tried it, you do let your children have a go - yep, it's fun for adults too!!!

Title:

TECHNO NINJA

Sold at: Micro Discount (see above for address)

Price: 4 pounds 95p (+ p&p)

Techno Ninja has been around for some time now so I thought a review was in order.

It turns out that you play a character

REVIEWS

called Garagon from the planet Trivator. Since a magical stone yielding infinite energy has been stolen from your planet, you are sent by the Techno Ninja's to find another one from the alien world of Sarcendor. [Yep, I think the members of Ke-Soft worked over-time to fill this game with names which take reviewers longer to type as well!].

The game takes the form of a platform and ladders-type game with the added bonus of your character being able to use weapons. Initially he starts with only his fists but can later gain other weapons such as a knife, sword, electro chain and the strangely named "Kinetic Gun"! Using these weapons can kill any baddies nearby or can demolish sections of a wall to pick-up any goodies lying behind them.

Speaking of the goodies, these are often needed in the form of keys to open doors or in the form of hearts to replenish your limited energy levels.

Graphically, Techno Ninja is in Ke-Soft's usual style - not especially brilliant but they do the job. On the music front, a good tune plays in the background. I must admit that the first time I played Techno Ninja (some months ago now), I was not too impressed with it. However, when I booted it up for this review, I couldn't get enough of it - it really is good fun!

Although Techno Ninja is no graphical treat, playability is where it counts and this game's got playability coming out of it's ears. Don't miss it!

Title:

TANKS

Sold at: Micro Discount (see above for address)

Price: Unknown (Ring for details).

TANKS is one of MIRAGE's latest releases, strongly based, I believe, on a game available for the PC.

The object of the game is to destroy enough of the enemy's tanks in order to advance to the next level. The game takes place in a maze-like screen (an overhead view) containing indestructable walls + a whole host of other items which randomly appear.

REVIEWS

For each square that your tank moves, it's oil level decreases by one. The only way to increase the level of this is to collect the oil power-up which randomly appears on the screen. Of course, you've got to beat the other tanks to it!

In a one-player mode, you are attacked by three other computer-controlled tanks. If you're longing for simultaneous two-player action, TANKS has an option included to allow you to play aginst a friend. Just remember that they are likely to be better than the computer controled opponents and are also more likely to hunt you down! You even get two computer controlled tanks on screen as well!

When enough TANKS have been destroyed, a doorway opens up for you to drive into, taking you to the next level.

One big let-down in TANKS is that the on-screen text (eg. your oil level) is not in English. Whilst you can work out that PALINO is oil, some of the words are prehaps destined to remain a mystery! Perhaps Derek Fern may have corrected the problem and supplied a translation sheet by the time this review is published.

Unfortunately, although the game is initially fun, shooting tanks until they explode soon becomes monotonous and you start to wish that the tanks weren't so heavily armoured!

Overall, I find it difficult to recommend TANKS. If you are looking for a two-player game and have a lot of patients, perhaps it's worth a purchase. If you don't have the patients, try T-34 TANK BATTLE for a better alternative.

Again, sorry for the reviews being slightly shorter again. Until next time, waggle those joysticks and tap those fire buttons and keep trying to repeat that high score!!!

TWAUG's new Postal Address:

TWAUG
c/o J. Matthewson
80 George Road
Wallsend
Tyne & Wear
NE28 6BU

WHY an ATARI

by Tommy Wood

It must have been about 1983 when everyone at work was raving on about ZX81 Spectrum Computers.

Not much later I was in the local Dixons looking at the aformentioned when I noticed a Demo on a TV of several tubes in different shades of gold and thought "thats good" what machines done that.. ATARI of course, I'll have that, thought I, until I found out the price £450 for an 800 48K machine and tape deck. I was hooked though compared to the Spectrums (£100 I think) there was no contest.

Several months later it was all mine. Many hours of typing in programs from the weekly computer magazines of the day, Page 6, (Now New Atari User) Computing Weekly, Analog and Antic and I was well and truly hooked. A computer Junkie!!

A few years later a 1050 disk drive at £200. 1987 saw the arrival of an Epson 9 pin printer at £150 and a 130XE for £45 and and in 1994 a 800XL upgraded to 256K.

In 1991 I received a telephone call from a man with a strange accent and a foreign sounding name,this turned out to be none other than MAX who lives locally and was looking for Atari Computing fanatics like myself. He in turn introduced me to the Tyne and Wear Atari User Group. We have been firm friends ever since.

In 1994 I decided I needed a Monitor instead of my old Tv set and saw an advert for a 520 STFM and a Philips CM8833 Mk.II Monitor for £190 bought the system and upgraded to 2 meg and a Panasonic 24 pin printer and now my setup consists of:

- 800XL upgraded to 256k. Two 1050 disk drives Eposon 9 pin printer. Philips Monitor.
- STFM upgraded to 2 meg. External drive 40 meg Hard drive Amstrad Color Monitor Panasonic 24 pin printer.

I use both machine for Wordprocessing keeping a database and weekly accounting and playing the odd game or two when I have the time.

Hope this will be of interest to your members.

VANISHING MAGAZINES

by Max

The year 1996 wasn't what you can call a very good year for the Atari computer. We saw two magazines disappear from the newsagents shelf, first, "The Atari World" ceased publication in March this year and that was followed by "STFormat" in August. The only magazine left for the Atari ST is an A5 size non-glossy mag. called "ST Applications" from FaST Club.

I believe the reason for that was lack of support, due to pricing themselves of the market. I know, and I am sure everybody else does, that a nice glossy magazine looks great, but the cost has to be high to keep it alive.

We've seen the same thing happening to "Page 6 New Atari User", it stopped producing a glossy magazine and went from an A4 size down to an A5 non-glossy issue.

Look at the TWAUG mag., right from the beginning it was plain in an A5 size type booklet with a coloured front and back page and it was readable, I think! Wasn't it? Well it must have been, because we did have a number of letters telling us that it looked good.

The reason I am writing this is to remind all subscribers that only with your support can we stay afloat. The only time we will go bust is when we loose our supporters. It wont be because of the publishing cost, we are doing the publishing ourselves.

There is another factor for ceasing publication and that is the lack of "MATERIAL", you see, we only put in the newsletter any material we get supplied and requests readers would like to read about. But lately no requests have been received and no new material was forthcoming. What I had to do in place of new material was to choose articles I thought would be of interest to our members.

After I've printed the master copy of the newsletter I usually wait in anticipation hoping that John would phone me with news that he received material or mail for the next issue, but that call never arrived lately. It is dissapointing not receiving any feed back, but as long you are all pleased with the articles in the issues, my enthusiasm for the Atari Classic and sitting up late into the night to publish TWAUG's newsletter, will not diminish.

What would be very disheartening for me is the disapearance of the TWAUG's newsletter. Prior to 1993 when TWAUG started this newsletter I hadn't done anything like this, it was a challenge and I enjoy it

VANISHING MAGAZINES

immensely. We've been going now for nearly four years, only one more issue to go.

If I would have to stop now, I would be completely at a loss. What could I do in my spare time? After I've done the gardening and the garden is quite large, done my house work, being a widower I've got to do it myself, that includes cooking, washing, shopping and some weeks baby sitting and only occasionally watching TV, what else could I do?

As I said I only watch the TV occasionally and that's only for the news, I am interested in world affairs, and also watch certain films only. Anything else on TV is rubbish and if I had to watch that kind of stuff it would really send me round the bend and I would end up with mad cow desease. As I'm not as young as I use to be and I never will be again either, that's why I'm saying, I'll get mad cow desease, when I stop doing what I enjoy. Don't take it seriously what I say, it's meant as a joke. So help us to keep TWAUG afloat, the newsletter that supports the 8-Bit Atari Classic.

Long live the ATARI CLASSIC

BIT WISE

Part One of MIKE BIBBY's easy-to-follow series looks at the number system at the heart of your Atari

WELCOME to the first in a series of articles in which we hope to take the mystery out of understanding the fundamentals of the Atari's workings.

All too often even competent Basic programmers tend to shy off such topics as binary coding, hexadecimal and assembly language because it seems too "mathematical".

This is a great pity, because the Atari is so constructed that a little knowledge in these fields allows you to take full advantage of its advanced facilities.

The mathematical aspects of the subject aren't at all deep -certainly anyone who can follow Basic should be able to cope with this series.

First we are going to look at binary code - a way of handling numbers essential to our understanding of what goes on inside a computer.

BIT WISE

It all adds up

|Binary is just a way of coding numbers in a way particularly suitable for computers. It's actually quite simple. What often confuses beginners is the fact that the binary system codes numbers in a way that can look extremely like the way we normally code numbers.

For example, if you were presented with the number 100, you would probably decode it in your normal way and say it was "one hundred".

That, however, is just one way of interpreting it. If you decided to decode it as a binary number, you would interpret 100 in a completely different way and say it meant the number "four". (Never mind exactly how you arrived at that conclusion for the moment.)

This is what often causes problems - people are so used to dealing with their numbers in the normal way that 100 is always "one hundred" to them, and they can't make the shift necessary to decode it in binary as "four".

Actually it is rather ambiguous.

to binary code

Presented with 100, do you interpret it as "one hundred" or "four"? Our rule will be, if you mean our usual way of dealing with numbers (the hundreds, tens and units you learnt at school - or put it more formally, the denary system) you write the number in the normal way.

If you wish the number to be decoded as a binary number you put the symbol % in front of it - 100 means "one hundred" while %100 means "four".

So far so good. We now have a marker (%) to warn us that we have to decode the number in a special way as a binary number.

However, before you decode you need a rule for decoding - so how do you get the number "four" from %100? What's the rule?

Let's take a detour for the moment, and think about the coins we use every day. Our currency consists of these coins:

50p, 20p, 10p, 5p, 2p and 1p.

We also use the 1 pound coin I hear

BIT WISE

you say, forget about that 1 pound coin here we are dealing only with the pence coin.

We can combine them to give any sum we wish: For example:

75p is 50p + 20p + 5p

or 50p + 10p + 10p + 5p

and so on. We are familiar with this - often we use multiples of coins to make up a sum. For example, 5p can be 2p + 2p +1p.

Using the same coin twice, though, often means that we end up carrying unnecessary amounts of change.

Sometimes, however, with our present coinage system we have to use the coin twice to obtain certain sums. You cannot, for instance, make up the sum of 4p without doubling up on coins. To avoid repeating coins we would have to invent a 4p coin!

Let's do that; in fact, let's invent a coinage system where you never have to use the same coin twice.

First of all we would need a 1p coin and, of course, a 2p coin, because we cannot use 1p + 1p for 2P - it breaks the rules!

Now 3p can be made up of 1p + 2p, but for 4p we'll have to invent a 4p coin.

Equipped with that we can make 5p (4p + 1p), 6p (4p + 2p), and 7p (4p + 2p + 1p). In obtaining 7p we used all our available coins, so now we have to invent an 8p coin. If you work it out (and I suggest you have a go) you will find that with the coins you have at your disposal (8p,4p,2p,1p) you can make any sum up to 15p. Then you would have to invent a new coin, 16p.

Notice that the coins we have created have double in value: 1p, 2p, 4p, 8p, 16p. No prizes for guessing what the next one is.

Let's summarise our results in a table (Figure I). Here I have used the columns to show the coins available and the rows to show how the various totals are made up. A 1 in a particular column means that we use that column's coin, and 0 means that we don't use it. Look at the row for 5p. It has 101 on it. According to our rule, this means we pick out the coins 4p and 1p (and NOT 2p) to make up the 5p total.

	4p	2p	1p	
%	1	0	1	
---►	4p	+	1p	= 5p

Now let's get back to computers by dropping all this talk about coins and redraw Figure I to show the same

BIT WISE

information but without referring to money - just numbers. Figure II is the new table.

As you can see, there is little change, and we can use this table to encode numbers in general, not just coins. We call this method of encoding the binary system.

Remember, to show that we mean a binary number we precede it with %. So if you see, for example, %101 means:

```
      4   2   1
%     1   0   1
--->  4   +   1 = 5
```

that is we add together the values of the columns containing 1. Look at row 5 of the table to check it. Similarly, %1101 would mean 13 in the denary system since

```
      8   4   2   1
%     1   1   0   1
--->  8 + 4   +   1 = 13
```

By now you should be able to work out for yourself why %100 represents four. From the table, or by using the addition method I've just illustrated, see if you can decode the denary values of the following binary numbers:

%1001

% 101

% 11

%1101

% 111

You can use the program accompanying this article to check your results. You've probably noticed by now that in the binary system you only two symbols, 0 and 1, to encode numbers - hence binary, bi - for two as in bicycle.

You can encode any number that you want in binary - just use more columns (or "bits" as we say in computer jargon), remembering that each new bit is worth double the preceding bit.

However, it does get terribly cumbersome. For example, 100 (denary) encoded in binary is %1100100 since:

```
      64  32  16  8  4  2  1
%     1   1   0   0  1  0  0
--->  64+32+4=100
```

It is much easier to handle the number in our normal system. To a computer this presents no problem, and the fact that binary only uses two symbols is a bonus because you can represent numbers with a sequence of "switches".

Switches are what we call "two state" - they're either ON or OFF. If

BIT WISE

	COINS			
	8p	4p	2p	1p
1p				1
2p			1	0
3p			1	1
4p		1	0	0
5p		1	0	1
6p		1	1	0
7p		1	1	1
8p	1	0	0	0
9p	1	0	0	1
10p	1	0	1	0
11p	1	0	1	1
12p	1	1	0	0
13p	1	1	0	1
14p	1	1	1	0
15p	1	1	1	1

TOTALS

Figure I

we have a sequence of four switches together we can encode numbers by having them either ON or OFF. We could use ON to mean a 1, and OFF to mean a 0 in a particular column:

	8	4	2	1
	ON	OFF	ON	ON
--► %	1	0	1	1 = 11

Each of these "switches" represents a bit, and a computer memory is full of bits. The 6502, which is the microprocessor at the heart of the Atari system, deals with many thousands of them.

To make things simpler the 6502 handles the bits in groups of eight bits at a time - the group of eight being called a byte.

With this type of organisation the largest number you can store in a byte is 255 since:

	128	64	32	16	8	4	2	1
%	1	1	1	1	1	1	1	1

---► 128+ 64+ 32+ 16+ 8+ 4+ 2 +1 = 255

BIT WISE

Denary Value	Column or Bit Values 8	4	2	1	Binary Value
1				1	%1
2			1	0	%10
3			1	1	%11
4		1	0	0	%100
5		1	0	1	%101
6		1	1	0	%110
7		1	1	1	%111
8	1	0	0	0	%1000
9	1	0	0	1	%1001
10	1	0	1	0	%1010
11	1	0	1	1	%1011
12	1	1	0	0	%1100
13	1	1	0	1	%1101
14	1	1	1	0	%1110
15	1	1	1	1	%1111

Figure II

Of course the computer can handle larger numbers (and not just whole numbers) but to do so it must use more than one byte.

Converting a byte from binary to denary is fairly straightforward. Simply write it down under the appropriate column (or bit) values and add together the value of all the columns in which a 1 occurs. For example, given %10010101 you translate as follows:

128 64 32 16 8 4 2 1
% 1 0 0 1 0 1 0 1
---► 128 + 16 + 4 + 1 = 149

Going from denary to binary is not at all difficult, but is rather hard to put into words. You do it by subtracting from the number you want to encode the value of each column in turn, starting with the highest (i.e. 128,64,32, and so on).

If you can subtract a particular

BIT WISE

column value you put a 1 in that column and continue to subtract the next lower column value from the remainder.

If you cannot manage the subtraction you put a 0 in that column and try to repeat the subtraction with the next lower column number.

So, starting with the highest column number (128 in our case), you:

1. Attempt to subtract the relevant column number (highest first).
2. IF you succeed THEN put a 1 in that column number and continue to subtract other columns from the remainder. Otherwise, put a 0 in that column.

Figure III should make it clearer.

In practice, when faced with encoding a number from denary to binary I tend to do it in my head, seeing which column values will add together to make the sum required, starting with the highest first.

For example, if I were to encode 161 in binary I would say, "Well, I canuse 128, so that leaves me 33 to find. 33 can be made up of 32 and 1 so that does it: 128+32+1=161. So I encode it as:

	128	64	32	16	8	4	2	1
%	1	0	1	0	0	0	0	1

= % 10100001

After a while you'll find this way quite simple.

To finish off, I'll leave you with a program to print out the binary value of a number between 0 and 255 (i.e. that can be stored in one byte). Try it with various values and see if you can accept the results.

The program itself uses one or two ideas that may not be too familiar to you as yet. Watch this space...

```
10 TRAP 190
20 GRAPHICS 0
30 OPEN #1,4,0,"K:"
40 ? CHR$(125)
50 POSITION 5,5
60 PRINT "Number";
70 INPUT NUMBER
80 IF NUMBER>255 OR NUMBER<0
OR INT(NUMBER)<>NUMBER
THEN GOTO 40
90 POSITION 5,10
100 FOR LOOP=7 TO 0 STEP -1
110 ANSWER=NUMBER-2^LOOP
120 IF ANSWER>-1 THEN
NUMBER=ANSWER:?"1";:GOTO 140
130 ? "0";
140 NEXT LOOP
150 POSITION 5,20
160 ? "ANY KEY FOR NEXT
NUMBER"
170 GET #1,DUMMY
180 GO TO 40
190 CLOSE #1
```

BIT WISE

149	
-128	128 goes - set to 1
21	64,32 can't go - set to 0
-16	16 goes - set to 1
5	8 can't go - set to 0
4	4 goes - set to 1
1	2 can't go - set to 0
-1	1 goes - set to 1
0	

128	64	32	16	8	4	2	1
1							
	0	0					
			1				
				0			
					1		
						0	
							1
%1	0	0	1	0	1	0	1

Figure III

FOR SALE

Black Box with Floopy Board, 90 Meg Hard Drive, 4 PBI Drives plus Modem all for £400

Contact John Matthewson on 0191 - 262 6897

CODING CAPERS

Hello there, bet you thought this column came to an end! Yeah well, I'm a pretty busy person at the moment and I don't get much time for home-computing at present. Actually, this is why I wrote this article, you see, a little while back I was supposed to write a DEMO for the British demonstration disk but never got round to it, very sorry lads to have let you down, but wait! Although I didn't actually get a DEMO together, I did write a lot of routines I was going to use, so guess what? I've sent all those routines to Max at TWAUG and he's given them to you. The routines are all seperate DOS files named CC01.OBJ through to CC23.OBJ. I've listed the filenames along with a small explanation in the table below:

CC01 - 4COLBAR+VERT-CLK

CC02 - 4COLBAR+PMG-BEND

CC03 - 8COLBAR-STNDRD

CC04 - 8COLBAR-PATTERN

CC05 - 16COLBAR/8LUM-STNDRD

CC06 - 16COLBAR/8LUMMIX-HLD/RLSE

CC07 - 32COLBAR/2-3LUM

CC08 - PMG ANGL-TXT-SCRL

CC09 - PMG WAVER+CLK

CC10 - 1PMG/2IMAGE/SHADOW-AFFECT

CC11 - PMG WAVER-STEP-PIECER

CC12 - GR.0 SINEWAVE ON GR.8

CC13 - GUNSHOT DIGI+SPL PLAYER

CC14 - GUNSHOT DIGI+MUSIC

CC15 - GUNSHOT+HI-THERE DIGI'S DBL-SMPL

CC16 - GTIA WAVER-MULTILAYED DM(OFFSET)

CC17 - GTIA WAVER-LINKED MULTILAY DM

CC18 - 3*3 CHAR-TXT-SCRL

CC19 - DBL-PLAYFIELD/OPPOSIVE SCROL

CC20 - PMG STARS (FROM TKS SNDMONITOR)

CC21 - GREETING+MESSAGE SCRL OVERLAY

CC22 - 6*45 PIX-TXT-SCRL

CC23 - PRESET WAVEFORM TXT-SCROL

You'll have to load the files seperately from the DOS menu option L. If any of them don't work properly, such as file CC23, then press RESET and wait. Right then, it's all up to you now to load up the files and see exactly just what I coded, below you'll find some

CODING CAPERS

general chit-chat about them:

CC01 to CC07: These 7 files all revolve around the use of my coloured-bar routine, they just show how many bars you can have, sizes, colours and colour-patterns, even movement alterations. It's quite surprising what you can do with just a simple routine.

CC08 to CC11: 4 files clown around with a Player-Missile Graphic. The 1st just shows an angled scroll, while the other 3 show different versions of using a PMG waving routine.

CC12, CC16 and CC17: You'll probably find it unexpected, but files CC16 and CC17 actually use the same routine as the sine-wave scroll CC12. I just made a couple of alterations, the graphic mode being one of them. Actually, these GTIA barbender routines have a few simple tricks. The biggest trick used is the repetition of DM to save processing time. You may also notice that there are tiny glitches in the make-up of the picture, this is due to picture alterations simmultaneous to flyscan activity. The cure to this should you be curious is used in file CC23.

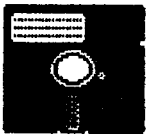
CC13 to CC15: Here's a digi-sample and a player I was doing some editing with. CC15 plays 2 digi's, the only difference to this file as to file CC14 is that the screen is turned off to gain the time to play 2 digi-channels as opposed to 1.

CC18: This 3*3 byte scroller is just 1 method (and a common one at that) as to scrolling text. It is a good method, but limitations are found with fonts designed with a decent selection of characters, especially regarding width. If you were to design your font in graphics then the processing time will incredibly tenfold or even hundredfold, but if your after fonts greater than 4 or 5 characters in width and still retain coloured detail then see the solution in file CC22.

CC19: This is probably the easiest method of opposing scrolls, by showing either one on opposite tv frames, though you'll have to keep the luminances low to reduce flicker, or don't put too much display on.

CC20: This is the stars affect taken from the TKS Soundmonitor, not programmed by myself. Them dots are PMG's.

CODING CAPERS continues on Page 35 ⟶



DISK CONTENT

Here we are again with a good selection of programs for you to enjoy.

Side A contains five programs all in Basic, easier for you to look into the program if you find it difficult to opperate.

The first game is a text adventure. When you run this game by pressing the appropriate number, the menu remains on the screen I have no idea why but it does. The text for the adventure is near the bottom of the screen.

The secong program on the menu is a utility to delete files on the disk without formatting.

TILES; this is the thlrd program on the menu. I must admit I am unable to say anything about it, I couldn't work out what it is about, there are no instructions with any of the program on Side A.

Now the fourth game on the menu is a different kettle of fish. I managed to play this game and got a score of 497 with the first attempt and by the time I'd lost all lives. Hurra! I am a champ. You are a stunt rider on a motor byke, on the novice level, the obstacles are jumping through a loop and over cars. Once the first lap is completed safely the distance from the ramp to the loop increases and another car is added for you to jump over.

The fifth game is called MARBLES, you must catch them in potts according to the colour, there are two colours red and blue.

HEAVENLY GATE, the last game on Side A, again I haven't work that one out, that means you have to find out yourselves how it plays.

Boot Side B with Basic, when loaded you are presented with the option of making a hard copy of all the doc files for the programs on that side of the disk, or read the instructions of the screen. These four programs are all utilities, and they are taking from our PD library.

With the instructions on the disk for all these programs, I wont have to go into a detailed description of the programs.

Enjoy!

ADVERTISING USER GROUPS

LACE

The London Atari Computer Enthusiasts

As a member of LACE you will receive a monthly newsletter and have access to a monthly meeting. They also support the ST and keep a large selection of ST and 8-bit PD software.

The membership fee is
£8.00 annually
for more information contact:
Mr. Roger Lacey
LACE Secretary
41 Henryson Road
Crofton Park
London SE4 1HL
Tel.: 0181 - 690 2548

O.H.A.U.G.

The OL'HACKERS ATARI USER GROUP INC.

O.H.A.U.G. is an all 8-bit user group in the STATE of NEW YORK.

They are producing a bi-monthly double sided disk based newsletter. The disk comes with its own printing utility, which lets you read the content of the disk, one screen page at the time, and/or you can make a hard copy of the disk, in one, two or three columns and 6 to 8 lines to the inch. A large PD Library is available.

Contact:
Mr.Ron Fetzer
O.H.A.U.G.
Secretary & Treasurer
22 Monaco Avenue
Elmont, N.Y. 11003
USA

CODING CAPERS

CC21: Again, using the same technique as CC19, the flicker method sometimes comes in handy. You can get away with the flicker by calling it something as though you planned for it to flicker, crafty trick eh!?

CC22: This is the big-scroll I created a little while back. It uses a different method to file CC18, but the processing time isn't that much greater in comparison to the font size. The advantages also are bigger characters and you can make them pictured and colourful. The amount of characters you can have can also be limitless, although in this case I only coded for 48 whilst having the font 6 bytes wide. The only snag with this method of font displaying, is that the deeper the characters, the more page-0 memory the routine requires.

CC23: This is my preset waveform scroller. The routine will scroll text through over 900(I think) x,y plotted points. Of course, the waveform pattern can be altered, just design your new one and put the x,y points into the file at the correct place. Actually, the scroll takes up a very lot of processing time (several(!) frames if I remember right), thus, I had to work out a technique to perform the character-plotting, zero-filling and moving without obtaining display-flickers due to the flyscan crossing over old-display positioning and new-display positioning. But how is this possible especially when the routine to just display the characters takes more than the whole frame. AHA! This is where this special technique flickers into the imagination. Actually, it's all to do with DMA. The DMA is used to my advantage by swapping screens alike page-flipping.

Anyhow, if I have any time for other things then I shall write further articles, but let's just see how it goes....

Tara for now, your friendly TOMOHAWK

The Routines for CODING CAPERS will be on the next issue disk.

We are sorry for the inconvenience caused, but the disk had already been prepared before we received the material from TOMO.

MICRO-DISCOUNT

Offers

The complete Mail Order service for Atari 8 Bit XL/XE users

4 th September 1995