# TYNE & WEAR

TM

ATARI 8 BIT

# USER GROUP

Issue 20

March/April 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.

The Atari Fuji symbol and Atari name are the trademarks of Atari Corporation. The Fuji symbol on the front cover , is for informational purpose only.

TWAUG is entirely independent and is in no way connected with Atari Corporation or any associate companies.

I have no doubt, you will have noticed the new style newsletter. For the last three and so years, from issue #1 and up to issue #18, the newsletter was printed with a Citizen 120D a 9 pin printer, it only produced near letter quality. My aim was to print the newsletter in a letter quality format, using Daisy-Dot and a 24 pin printer. So I purchased a Star 24 printer, but instead of printing in letter quality, the fonts were elongated and looked distorted, that was no good and so I continued printing with the 9 pin printer and my new 24 pin had to stand there, doing very little work.

It was last year, 1995, when a friend introduced me to an Atari ST, I was very impressed with the performance. He showed me what a DTP program can do, I didn't realize that the ST machine was so good, I was never interested in anything but the Atari Classic, I have a real good setup of the old 8-bit.

But, after seeing what a Desktop publishing program can do, made me think. There is a lot of work involved setting up the columns of a newsletter, using Daisy-Dot. To make life a little easier I decided there and then to invest in a ST and so I got a Mega 1 ST and shortly after that upgraded to 4 meg.

It took me awhile to get acquainted with the machine, playing about with it in between working on the newsletter and assembling some utilities, but I am now ready to use the ST for the newsletter. Issue #19 was the experimental issue, I found the setting up of the newsletter much easier and faster and at long last I've got the letter quality print I was looking for.

### Feed Back

Give us your views on the new look newsletter, do you find it easier to read and do you like the smooth typesetting better than the old one?

**TWAUG**

**PO BOX 8, WALLSEND**

**TYNE & WEAR , NE28 6DQ**

# TWAUG NEWSLETTER

## CONTRIBUTIONS!

We need some help from our subscribers, we would like you to submit some articles for publication. We welcome our readers to submit, articles, reviews, programs and views for publication. We cannot make cash payment, but you can have an issue for each article or choose from the PD Library. Help us to keep TWAUG alive.

**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 21 is due for distribution by mid-May**

## ISSUE CONTENT

# DON'T LET BASIC BUG YOU

**Expand your knowledge of programming with PART THREE of MIKE BIBBY's guide through the micro jungle**

We saw last month how to write our own programs, however primitive. Now we'll look at some ways of improving them. I don't guarantee that you'll be able to produce spectacular programs by the end of this article, but you will certainly be well on the way to an understanding of Basic.

First, though, let's recap a little: We saw last month that a Basic program consists of a numbered sequence of instructions to the computer.

To enter one of these instructions we simply type the correct line number, followed by the appropriate Basic keyword, then press Return.

As we discovered, because of the line number, the Atari doesn't do what you tell it immediately, but remembers it as part of the program.

To see all the instructions in a program, we type:

LIST Return

To actually get the Atari to carry out the sequence of instructions we type:

RUN Return

To clear a program from memory (and we should do this before entering a new program), we use:

NEW Return

We saw that we tended to enter line numbers in steps of 10 to allow us to fit in other instructions between them if necessary. Also we found that we could replace a line with a better version by simply giving the new version the line number of the old one.

## String along......

```
10 PRINT "PROGRAMMING"
20 PRINT "IS"
30 PRINT "EASY"
```

Program I

Finally, to delete a line completely, we simply type the line number and press Return.

## and pick up

Program I is the one we started with last month. Before we continue, type it in and run it, to make sure you know what's going on.

Program II is another way of achiev-

## DON’T LET BASIC BUG YOU continued

ing exactly the same output. Type it in and try it.

```
10 DIM A$(12),B$(12),C$(12)
20 A$="PROGRAMMING"
30 B$="IS"
40 C$="EASY"
50 PRINT A$
60 PRINT B$
70 PRINT C$
```

Program II

Apart from being an incredibly long-winded way of doing things, what else is going on?

Well, as you will recall from the first article in the series, the words inside quotes are known as strings - because the computer simply remembers them as stings of letters. That is, it considers HAMPSTER as H, followed by A, followed by M and so on, with no idea of the word’s meaning.

## some handy jargon

I don’t think that it takes all that much imagination to see that when your computer is printing a lot of output, you might be using the same string rather a lot.

For example, in a business letter you might use the name of the company fairly frequently - for instance, BBC for British Broadcasting Corporation. Atari Basic allows us to use much the same idea, but more as labels than abbreviations.

For instance, in line 20 of the above program we have labelled the string “PROGRAMMING” with the label A$.

In computer terms, we have assigned to A$ the value “PROGRAMMING”.

All this means is that from now on wherever I want to use “PROGRAMMING” in my program, I can replace it with A$. So line 50, which is:

**50 PRINT A$**

causes the micro to print out “PROGRAMMING”.

Admittedly in this example this technique of labelling doesn’t save much space or effort, but if the program uses the word “PROGRAMMING” 100 times, there would be a substantial saving in using A$ instead of the string itself.

Similarly, line 30 causes B$ to label IS and line 40 labels EASY with C$, so that lines 60 and 70 give the appropriate printout.

Notice the following points:

# DON'T LET BASIC BUG YOU continued

- We have chosen our labels so that they consist of a letter of the alphabet followed by the $ sign. Actually, we don't have to restrict ourselves to just one letter, as we shall see, but our label must end with the $ sign, since this warns the computer that we are labelling a string. And the letter we use must be a capital. (We'll see later how to label other things.)
- While I used A$ for the first label, B$ for the second and C$ for the third, this was totally arbitrary on my part -labels don't have to follow alphabetic or any other kind of order.
- Although we use an equals sign (=) to connect the label with what it is labelling, it is safer, as we shall see, not to think of it as an equals sign - think in terms of A$ becomes “PROGRAMMING” rather than A$ equals “PROGRAMMING”.
- We must have the label on the left and what is labelled on the right of the equals sign. A line such as:

```
20 “PROGRAMMING” = A$
```

just does not make sense to the Atari. Try it for yourself!

- When labelling we put the string inside quotes, as we did previously when using the print statement to print out strings. So line 20 reads:

```
20 A$=”PROGRAMMING”
```

From now on A$ completely replaces “PROGRAMMING”, quotes and all, so that when we say

```
PRINT A$
```

we don't have to use quotes - they're already there, implicit in the label A$.

All right, but we still haven't explained line 10:

```
10 DIM A$(12),B$(12),C$(12)
```

Well, it's all to do with the micro's good housekeeping. Just as, when you throw a party, it's helpful to have an idea of the maximum number of guests you expect, so it's only manners to tell the Atari how large you think each string to be. It can then set aside a suitable amount of memory for the strings.

We do this with DIM - a new Basic keyword that fixes the maximum number of letters or characters to be associated with each label.

For instance, if we had a string label X$ and we never wanted it to refer to a string of more than ten characters in length we would have a line such as:

```
10 DIM X$(10)
```

# DON'T LET BASIC BUG YOU continued

**Notice:**

- The keyword DIM followed by a space.
- The label X$ followed directly - no space - by the maximum length you want to label, in brackets.

That's what we did in line 10 of Program II. This time we had three labels to dimemsion - A$,B$,C$ - so we put them all in the same line, separated by commas.

You might also notice that I've been pretty wastefull with my dimming, as it is known - I've given each label a maximum length of 12, although, as you'll see from the rest of the program, none of my strings is that long.

I could have got away with:

**10 DIM A$(11),B$(2),C$(4)**

Try running Program II with this new line, if you don't believe me. Remember, all you have to do to alter a line is to retype it (starting with the line number of course), then press Return. The new version of the line will replace the old one.

What would happen if we didn't DIM enough room for a string being labelled? Try replacing line 10 with:

**10 DIM A$(8),B$(2),C$(4)**

If you've done it properly, when you run it you should get the message:

**PROGRAMM**

**IS**

**EASY**

As you can see, the label A$ accepted as little as possible.

All right, but you wouldn't make this sort of mistake, would you? After all you can just look at a program and see how big the strings you're labelling are going to get.

Yes, but the strings you're labelling can change size as in Program III, where what X$ labels varies from BIG via BIGGER to BIGGEST. Hence another, more common name for these labels - string variables.

```
10 DIM X$(7)
20 X$="BIG "
30 PRINT X$
40 X$="BIGGER"
50 PRINT X$
60 X$="BIGGEST"
70 PRINT X$
```

Program III

Notice each time you give a string label or variable a value, that value "replaces" the old value. These variables really vary.

## DON'T LET BASIC BUG YOU continued

Now when we label a string the label refers to whatever is inside the quotes, including spaces, as you will see if you run Program IV.

Notice that our punctuation - semicolons - works for labelled strings just as it worked on its own.

Notice also that we have introduced a new Basic keyword in line 10 - REM. We use REM which is short for REMark, to add comments or headings to our programs.

When the Atari encounters REM in a line it ignores everything else after it on the same line. This means we can write whatever we want after REM (providing it is on the same line) without fear of the micro giving us an error message - the Atari doesn't "read" the line beyond the REM.

**"It is good programming practice to include REMs"**

If we use REM to prefix our comments, we can annotate our program. Certainly each main subdivision should have one or more REM statements explaining what is going on.

Since the Atari ignores the contents of REM statements, you could leave them out of your program entirely and it will work as effectively. However, it is good programming practice to include them.

In Program IV I have used a single REM at the beginning of the program, as it is so short. Bear in mind however, that REM can appear on any line in a program.

Now for some jargon. From now on we shall refer to our labels as variables. Don't be put off by the mathematical sound of that - they are still just labels. And instead of saying we are labelling, we say we are assigning, as we have mentioned previously. The actual string involved is known as the value of the variable. So:

A$="TEST"

```
10 REM PROGRAM IV
20 DIM A$(7),B$(7),C$(7),D$(7)
30 A$="TEST"
40 B$=" TEST"
50 C$="  TEST"
60 D$="   TEST"
70 PRINT A$;B$;C$;D$
80 PRINT "01234567890123456789
0123456789"
```

Program IV

# DON'T LET BASIC BUG YOU continued

reads "the string variable A$ has assigned to it the value TEST". The actual act of giving a variable a value is called an assignment.

To return to the world of actual programs, you can mix and match string variables and actual strings however you want.

Program V illustrates the point:

```
10 REM PROGRAM V
20 DIM A$(10),B$(5)
30 A$="MY NAME IS"
40 B$=" MIKE"
50 PRINT A$;B$
60 PRINT "MY NAME IS";B$
70 PRINT A$;" MIKE"
```

Program V

Notice the space at the beginning of the string assigned to B$ -you need this otherwise the output looks rather odd. Leave it out if you don't believe me.

As we saw last month, a semi-colon at the end of a line causes the next output to start immediately after the last and not on a new line - as it would do in the absence of the semi-colon. That is, it "glues" the strings together.

The internal semi-colons of lines 50, 60 and 70 do much of the same, "gluing" variables to strings, and so on.

Also, on the subject of grammatical propriety, when we're assigning variables we should use the LET statement. So line 40 should read:

**40 LET B$="MIKE"**

As you've already discovered, we can omit LET altogether.

Next month, more on variables and INPUT - which opens the door to effective programming.

## PAGE 6 WRITER INSTRUCTIONS

Page 6 Writer was written by Phil Cardwell and originally appeared in issue 50 of New Atari User as a type-in program. It was also on the issue 50 disk. Both the back-issue magazine and disk No 50 are available from:

**Page 6 publishing , PO Box 54,**

**Stafford ST16 1DR**

**Tel.(01785) 41153.**

The program is quite compact (only 67 sectors long) and surprisingly powerful for its size. This compactness lends itself to cassette use and will

# PAGE 6 WRITER INSTRUCTIONS continued

load in about 3.5 minutes from a standard (non-turbo) tape. These instructions have been compiled, using as a base the instructions for **'SPEEDSCRIPT'** which is a similar word processor; modifying them as necessary to match the documentation of the original article and adding comments as a result of experience with the program.

## Please note the following:

- **CTRL-H** means hold down **CONTROL** while typing **H** (upper case)
- **SELECT-h** means hold down **SELECT** while typing **h** (lower case)
- **SHIFT-*** same system but with SHIFT key.

## Loading the program

- **DISK**: hold down **OPTION** while switching on the computer.
- **CASSETTE**: hold down **OPTION** and **START** during switch-on.

When loaded, the monitor will show a black screen with a blue band at the top bearing the prog title and the author's name. When required during use this band (command line) will show prompts to instruct or warn you; some prompts will require a response from you and this will be indicated.

## Some observations.

A number of SPEEDSCRIPT's features, although not included in Phil Cardwell's article, do nevertheless appear to be present and working. Not everything has been tried however so it is recommended that you read the original write-up and try things for yourself. Some characters do not at first seem to be available. The backslash **(SHIFT-+)** is an example and this is fundamental to the **DAISY-DOT** series of programs; but if **ESCAPE** is pressed before **SHIFT-+**; Hey-presto a backslash! This system will no doubt work with other missing characters.

Holding **OPTION** while loading from disk is not necessary but the resultant disabling of **BASIC** makes more memory available for editing. Cassette versions will not load however, if **OPTION** is not held. On the subject of cassettes; Denham Phipps of Bradford has noticed that if **RESET** is pressed the screen goes blank and your text seems lost but if **BREAK** is pressed twice all is restored. This does not happen with disk based versions.

## John Bunting February 1994

## Command Summary

- **CTRL-+** = Move cursor 1 space left
- **CTRL-*** = Move cursor 1 space

## PAGE 6 WRITER INSTRUCTIONS continued

right

- **CTRL**- minus = Moves cursor backwards to beginning of previous sentence
- **CTRL-=** = Moves cursor forward to beginning of next sentence
- **SHIFT-+** = Moves cursor left to the beginning of previous word
- **SHIFT-*** = Moves cursor right to the beginning of next word
- **SHIFT**- minus = Moves cursor back to beginning of previous paragraph
- **SHIFT-=** = Moves cursor forward to the beginning of the next paragraph (*,-,= and + are the arrow keys.)
- **CTRL-H** (hit once) = Moves cursor to top of screen without scrolling
- **CTRL-H** (hit twice) = Moves cursor to start of document
- **CTRL-E** = Moves cursor to end of document
- **CTRL-insert** opens up one space
- **CTRL-I** = Typeover mode.

All text typed in this mode will overtype that at the cursor position; it is indicated by a black command line.

- **CTRL-I** again returns to the default (insert mode):

All text typed in this mode will be inserted at the cursor position with the following text moved to the right. It is indicated by a blue command line.

- **TAB** = Inserts 5 spaces for each operation
- **SHIFT-insert** = Inserts 255 spaces for each operation
- **SHIFT-Delete/Back S.** = Deletes all spaces between cursor and following text
- **DELETE BACK S.** = Deletes the character to the left of the cursor
- **CTRL-Delete Back S.** = Deletes the character on which the cursor is sitting
- **CTRL-D** = Erases text after (to the right of the cursor.

The message ''Erase **(S,W,P)**: RETURN to exit. Type **S** to delete a sentence, **W** for a word, or **P** for a paragraph. You can continue to press **S,W,** or **P** until all the desired text is deleted, then press RETURN to exit RETURN = In general allows an escape from any Page 6 Writer command

- **CTRL-R** = Restores the data removed by the **LAST CTRL-D** operation.

**********NOTE*************

To move blocks of text from one

## PAGE 6 WRITER INSTRUCTIONS continued

location in the document to another first erase or delete it with **CTRL-D** then move the cursor to where the text is desired. **CTRL-R** will insert the text (sentence word or paragraph) at this point.

To duplicate blocks of text use **CTRL-D** to erase text then **CTRL-R** to restore it. Move cursor to new location and **CTRL-R** to restore the text again at the new location.

A buffer is used for the **CTRL-D** function (2K for disk, about 6K for tape).

With this command you can erase up to the capacity of the buffer, but **CTRL-D** erases the buffer with each use. If you need to preserve the buffer contents, holding **OPTION** while pressing **CTRL-D** will preserve the buffer contents and add the newly erased text to it up to its full capacity.

- ♦ **SHIFT-Clear** or **CTRL-Clear** = Clear all text.
- ♦ Type **Y** or **N** at the "**ERASE ALL TEXT**" prompt.

## Search And Replace

**OPTION-CTRL-F** = Activates search feature. The command line prompts "**Find:**". Type in the search phrase (what you would like to search for). Typing **CTRL-F** again will search for the next occurrence of the phrase after the cursor position (press start twice to searsh entire document).

**OPTION-CTRL-C** = Provides a replace phrase to be substituted for all incidences of the search phrase.

Entering RETURN at the "**CHANGE TO**": prompt will delete all occurences of the phrase **CTRL-G** (Global) = Links **CTRL-F** and **CTRL-C** together prompting "**FIND:**" then "**CHANGE TO**" and replacing all occurrences of the search phrase.

********NOTE********

Search will find embedded words (such as "the" in the word then). To search for a single word precede it with a space. Use a space in the **REPLACE** word also.

### Storing and Loading a Document

- ♦ **CTRL-S** = Save.

Type **C**: or **D**: (legal filename) at the prompt. No Errors prompt appears after successful save.

- ♦ **CTRL-L** = Load.

Same procedure at prompt. No Errors indicates successful load.

********NOTE********

Files begin loading at cursor position, so be sure to press **CTRL-H** twice (moving cursor to beginning of docu-

# PAGE 6 WRITER INSTRUCTIONS continued

ment) or **CTRL-CLEAR** or **SHIFT-CLEAR** (Erase all) unless you intend to merge files. To merge files use **CTRL-E** to position cursor at end of text then load the file to be merged. **A LOAD** operation does not insert text but will overwrite any text after the cursor position.

## Programming with Page 6 Writer

Since Page 6 Wṛiter stores files in **ASCII** you can load any **ASCII** file with it. You could write a **BASIC** program with Page 6 Writer, save it on disk then use ENTER to read the file from BASIC. In BASIC you can store a program in ASCII form with **LIST"D:filename"** or **LIST "C:" FOR TAPE**. You can even load files produced by most other word processors those programs can read Page 6 Writer files.

## Disk Commands

- ♦ **CTRL-M** = Menu (Disk Directory.)

Puts disk directory on screen in three columns with a large cursor on current file. Use cursor arrow keys (no need to press **CONTROL**) to choose file, follow menu on screen using

- ♦ **CTRL**-L to load file,
- ♦ **CTRL**-D to delete,
- ♦ **R** to rename,
- ♦ L to lock,
- ♦ U to unlock,
- ♦ F to format the disk.

Changes to the directory will not be seen until the directory is called again.

**PLEASE NOTE!**

- ♦ **CTRL-D** will delete a highlighted file without warning.

## Additional Features.

- ♦ **CTRL-X** = Exchanges the character under cursor with the character to the right of the cursor.
- ♦ **CTRL-A** = Changes the character under the cursor from upper to lower case or vice-versa.
- ♦ **CTRL-Q** = Quit to **DOS**.

A warning is given on the command line -respond with '**Y**' if you are sure.

********NOTE********

Atari 400/800 owners will notice that the **CAPS/LOWER** key now functions as a toggle between upper and lower case; it works like the **CAPS** key on the **XL** and **XE** series. Press once to lock in upper case, again to lock in lower.

- ♦ **CTRL-B** = Background colour.

Each time **CTRL-B** is typed the background colour is changed to 1 of 128 different colours.

## PAGE 6 WRITER INSTRUCTIONS continued

♦ **CTRL-T** = text luminance.

Each time **CTRL-T** is typed 1 of 8 text luminances is cycled.

**WARNING!** The first press of **CTRL-T** turns the text black like the background making it disappear - repeat the procedure a few times to bring it back again.

♦ **OPTION-CTRL-+** = Decreases screen width.

Each keypress narrows screen width by one space from a maximum of 40 to a minimum of 2

♦ **OPTION-CTRL-*** = Increases screen width, reverse of above

♦ **CTRL-Z** = Provides a method of distinguishing between true spaces and false ones (those generated by a word wrap).

This command causes false spaces to appear as tiny dots. You can write or edit in this mode if you wish, or exit with **CTRL-Z** again.

********NOTE********

The break and inverse video keys are disabled by Page 6 Writer. If inverse characters are needed hold **SELECT** while typing the key.

********NOTE********

### PRINT!

If your printer incorporates an automatic skip over perforation feature turn this feature off before printing. Align the peforation with the top of the print head.

♦ **CTRL-P** = Print.

At the Print:(Device:Filename)>'' prompt, simply enter **'P'** RETURN to print to most printers. Enter **D**:Filename for disk, **C**: for cassette, **E** for screen if you wish to print to any of these.

♦ **BREAK** = Hold until printing stops to abort

♦ **CTRL-1** = pause printing.

♦ **CTRL-1** again to resume

********NOTE********

If you need to print to an **RS-232** printer, just Print to a disk file, then boot up your DOS master disk and use the copy selection to copy the print file to the **R:** device. You can also write **BASIC** programs to read and process a Printed disk file. Remember, a Print to disk is not the same as a Save to disk.

### FORMATTING COMMANDS

**Stage 1 Commands**

(Usually control variables such as margins, linespace, etc.) must be lower case.

♦ **SELECT-1** = Left margin.

Enter a number from 0 (no margin) to 255. Default=5

## PAGE 6 WRITER INSTRUCTIONS continued

- ♦ **SELECT-r** = Right margin.

Enter a number from 1 to 255. Default=75

- ♦ **SELECT-t** = Top margin.

Position of first line. Default=5.

The header (if any) is always printed on the first line of the page.

- ♦ **SELECT-b** = Bottom margin.

Position of last printed line before next page. Default=58.

Do not enter a number greater than page length.

- ♦ **SELECT-p** = Page length.

defaults to 69.(A4 size) If you use 11" paper the setting should be 66.If your printer does not space 6 lines to the inch, multiply lines per inch by paper length in inches for page length.

- ♦ **SELECT-s** = Line space.

Enter 1 to 255. Default=1

- ♦ **SELECT-%** = Starts numbering at page number given. Normally 1
- ♦ **SELECT-?** = Disables print until desired page number is reached.

A value of 3 would start printing at page 3 of the document. Normally printing starts with page 1. Put this at the start.

- ♦ **SELECT-x** = Sets page width.(Think a cross) Default=80.

Needed for centering in double width or condensed type or if using a 40 column or wide carriage printer.

- ♦ **SELECT-n** = Forced paging.

Stops printing then resumes at start of next page, but does print any footer.

- ♦ **SELECT-m** = Margin release.

Disables margin for next line.

- ♦ **SELECT-w** = Page wait.

Should be placed at beginning of document before any text. With page wait on you are prompted to "Insert next sheet, press RETURN" when each page is finished printing.

- ♦ **SELECT-j** = Select automatic line feeds after carriage return.

Like **SELECT-w** this command must be placed before any text. Do not use for double spacing, but only if all text prints on the same line.

- ♦ **SELECT-q** = Quotation.

Like a REM statement. Follow this command with a message up to 255 characters terminated with a RETURN. This line will be ignored during printing.

- ♦ **SELECT-h** = Header define and enable.

The header prints on the first line of each page. Header must be a single line

## PAGE 6 WRITER INSTRUCTIONS continued

of text up to 254 characters ending with RETURN. You can include Stage 2 commands, centering etc. in a header. You can use a header by itself without a footer. The header and footer should be defined at top of your document before any text. If you do not want the header to print on the first page, put a return mark by itself at the top of your document before the header definition.

- **SELECT-f** = Footer define and enable.

Instructions are the same as for the header. The footer prints two lines prior to the last line of the page.

- **SELECT - i** = include a file. (link next file.)

Put this command as the last line in your document. Follow it with the filename, including **D**: for disk. After the text in memory is printed, the link command loads the next file into memory.

### Stage 2 Commands

- **SELECT-c** = Centres one line.

Put this command at the start of the line you want to centre. to centre a double width line set the page width to 40 or pad out the rest of the line with an equal number of spaces.(See information under SELECT-x).

- **SELECT-#** = Prints current page number.

You usually embed this command within a header or footer.

- **SELECT-u** = A simple form of underlining.

It works only on printers that recognize **CHR$(8)** as a backspace and **CHR$(95)** as an underline character. Underlining works on spaces, too. Use the first **u** to start underlining, and another one to turn off underlining.

- **SELECT-e** = edge to right margin (right justify).

**SELECT**- (any upper case character not used for other commands) = ***Printkey***. To designate a ***Printkey*** hold the **SELECT** button and type the key you want to assign, then type = (*equals sign*), and finally the **ASCII** value to be substituted for the ***Printkey*** during printing. This allows definition of **ASCII** values which your printer uses for features like double width or emphasized mode. Four ***Printkeys*** are predefined by Page 6 Writer, but may be changed if desired. The keys 1 through 4 are defined as **27**,**14**,**15**, and **18**.

On most printers - **CHR$(27)** = **ESC**ape key

**CHR$(14)** = One line double width

**CHR$(15)** = Either ends double width or starts condensed characters

**CHR$(18)** = Usually cancels con-

# PAGE 6 WRITER INSTRUCTIONS continued

densed characters.

As an alternate method the key with the **ASCII** value of the Control code (ie Cont - **N** = 14 = Start double width) can be embedded in the text.

## Commands Alphabetically

## CTRL + ACTION

**A** Changes character at cursor from upper to lower case and vice-versa.

**B** Cycles through background colours.

**C** (Hold OPTION) see search and replace.

**D** DELETE - see search and replace.

**E** Cursor to end of document.

**F** (Hold OPTION) Find - see **S** and **R**.

**G** Global search and replace.

**H** Press once puts cursor to top of page. Press twice puts cursor to top of document.

**I** Toggles insert and typeover modes.

**M** Menu of D1; follow by 1-8 for directory that disk, which becomes default.

**P** Print - follow by 'P' for printer.

**Q** Quit to DOS.

**R** Inserts buffer contents at cursor position.

**S** Save to disk or cassette.

**T** Cycles through text luminence.

**W** Displays free memory.

**X** Swops cursor character with one on right.

**Z** Puts dots in false spaces, repeat to cancel.

| SELECT + ACTION | DEFAULT |
|---|---|
| **b** Bottom margin | 58 |
| **c** Centering | |
| **e** Edge right | |
| **f** Define footer | |
| **h** Define header | |
| **j** Select linefeeds | |
| **l** Left margin | 5 |
| **m** Margin release | |
| **n** Next page | |
| **p** Page length | 69 |
| **r** Right margin | 75 |
| **s** Spacing | 1 |
| **t** Top margin | 5 |
| **u** Underline toggle | |
| **w** Page wait off | |
| **x** Columns across | 80 |
| **#** Page number | 1 |
| **%** Starting page number | 1 |
| **?** Print starting page number | 1 |

# 130XE COMPATIBLE - 256 UPGRADE FOR THE 800XL

## SUPPLIED BY THE CHAOS BBS (517) 371-1106

A 130XE - COMPATIBLE 256K UPGRADE FOR THE ATARI 800XL
by Claus Buchholz

I designed the 256K upgrade described in my article, "The Quarter-Meg Atari" (BYTE, September, 1985 and recapped here in this article), in December, 1984. Since this predated the 130XE, there was no precedent for extended memory on the XLs. I felt free to implement a system of eight 32K banks. The major reason was to keep the add-on circuit as simple as possible.

The 130XE, introduced in early 1985, set a different standard for bank-select memory. It uses 16K banks and makes them separately available to both the CPU and the video controller (ANTIC). The XE has 128K total memory. The 64K extended RAM is split into four 16K banks. A 256K 800XL has 192K extended RAM, which requires 12 16K banks. I have designed a new upgrade for the 800XL that implements such a scheme. Its similarity to the 130XE's scheme allows use of software for the XE on a 256K 800XL.

To select one of four banks, the XE uses two bits, #2 and #3, in the memory control register (port B of the 6520 PIA, addressed at $D301 or 54017 decimal).

Zeroing bit #4 makes the selected bank appear at addresses $4000-$7FFF (16384 to 32767 decimal), as seen by the CPU.

Zeroing bit #5 makes it appear there as seen by ANTIC. In my upgrade, bits #2, #3, #5 and #6 select one of the twelve banks.

Zeroing bit #4 makes the selected bank appear at $4000-$7FFF to both the CPU and ANTIC.

So, any program for the XE that uses the extended RAM for CPU storage will work on an 800XL with this mod. Those programs won't use the additional 128K, though. Programs that use the video banking feature of the XE might run on the modified XL, but the screen display will be wrong.

## The Dynamic RAM

Each chip inputs or outputs one bit at a time, so each bit has a unique address. For the 256K-bit chip the address requires eighteen bits. The chip has nine address inputs, each of which does double-duty. During the first part of a memory access, half of the address bits are presented to the chip. This half is called the row address. Later in the access cycle, the chip receives the other half of the address, called the column address. The storage cells in the chip lie in a matrix, and the cell being addressed

## 256 UPGRADE FOR THE 800XL continued

lies at the intersection of the row and column specified. To complete the access cycle, the chip reads or writes the selected bit. The 800XL uses 64K-bit RAM chips, which have eight address inputs for an eight-bit row address and an eight-bit column address. This is fine for the 16-bit addresses the 6502 gives. There are eight of these chips, each contributing one bit to each byte of RAM. The 256K-bit RAM is practically identical to the 64K-bit RAM except that it has one extra pin to accommodate the two additional address bits it needs.

This extra pin is pin #1 on the chip. Pin #1 on the 64K-bit chip has no function. The functions of all the other pins on both chips are identical. Therefore, our upgrade involves unplugging the eight 64K-bit RAMs and plugging eight 256K-bit chips in their place.

We must also add some circuitry to provide two extra address bits for pin #1. The storage cells in dynamic RAM chips are actually microscopic capacitors, storing an amount of electric charge that represents a 0 or 1 bit. Since capacitors leak charge, they must be periodically recharged or refreshed. The chip refreshes one or two entire rows when accessed. This means that every row must be accessed frequently to keep the stored data accurate. Since normal operation of RAM can't guarantee that, the computer system must provide special access cycles called refresh cycles. A refresh cycle is a dummy read cycle in which a refresh address is used as the row address.

The 16K- and 64K-bit RAMs require seven-bit refresh addresses. The computer must provide all 128 possible refresh addresses every few thousandths of a second to keep the RAM refreshed. In the Atari, the video controller provides refresh cycles in addition to its screen memory accesses. It automatically provides seven bits for the refresh address. It turns out, the Atari spends four percent of its time refreshing RAM. One snag in designing the 256K-byte upgrade is that standard 256K-bit RAMs require an eight-bit refresh address. Older versions of the Atari video controller chip provide only seven bits of refresh address, whereas newer versions give all eight. So, there are two versions of the upgrade's interface circuit. The more complex one must add another bit to the Atari's refresh address.

### Notes on Bank-selection

To fit 256K bytes into the 6502's 64K memory space, we must divide it into banks. If a program in RAM were to replace the entire 64K RAM with another bank, it would cause itself to disappear, and the system would certainly crash. Also, the top 32K of the Atari's address space is cluttered enough with hardware addresses and ROMs that

## 256 UPGRADE FORTHE 800XL comtinued

can be switched in and out themselves. The screen RAM is usually in the top 32K and we don't want to switch that out and cause glitches to appear on the screen. Additionally, the operating system keeps important data in the lower part of RAM and it expects the data to be there when it is called. Further, the 6502's stack is in low memory. Worse, interrupts occur frequently and the routines they invoke also keep data in low RAM.

**Programs must therefore follow a strict rule:**

Keep the "normal" bank enabled as much as possible. If you select another bank, you must first disable all interrupts and not call the operating system until the "standard" bank is restored.

### Interfacing the RAM

The interface circuit for the 256K RAM is to be assembled on a small circuit board and installed inside the computer, as the computer's expansion slot doesn't carry the signals we need. The circuit consists of four (or five for older models) chips and replaces one of the chips on the computer motherboard. It also requires jumper wires to various points on the motherboard and connection to 5 pins of the PIA (U23). The circuit plugs into the socket at position U27 on the motherboard. This gives it access to six important signals, including power and ground.

The chip that was at U27 becomes IC1 in the circuit.

As U27, this chip was one of the two responsible for selecting which eight of the sixteen address bits are passed to the 64K RAMs at one time.

If your **ANTIC (U7)** part number is **CO21697**, use the circuit described by the first connection list at the end of this article. If it is **CO12296**, include the circuit in the second list. The circuit requires five connections to the **PIA (U23)**. So, pins **12 through 16** must be bent up and connected to the circuit. The extra circuitry for the older version of the 800XL is an eight-bit binary counter that counts the refresh cycles. It supplies the eighth bit of the refresh address that the 256K chips need. The refresh signal it uses comes to the circuit through a jumper wire from the motherboard.

### Performing the Upgrade

To disassemble the 800XL, remove the six screws on the underside and separate the top and bottom portions of the plastic case. Be careful of the flexible keyboard cable. Pull it straight up out of its socket on the motherboard. To detach the motherboard from the case bottom, remove three screws: one on the right side, one in the right rear corner, and

## 256 UPGRADE FOR THE 800XL continued

one in the left rear. Gently pull the board free of the case. Next, remove the small nuts and bolts around the metal shielding that encases the motherboard. On the left side of the exposed motherboard, locate the row of eight 16-pin RAM chips. Just to their right is **U27.** Behind **U27** is a three-inch square area that fits inside the shielding. The circuit goes there, because the shielding is highest toward the rear.

Replace the 64K RAMs with the 256K RAM chips. The new RAMs are very easily destroyed by static discharges, so extreme care is necessary in their handling. Lay aluminum foil on the work surface and keep the motherboard, RAM chips, tools and hands in contact with the foil at all times. This keeps everything at the same potential, decreasing the possibility of damage.

On the motherboard, locate the video controller, the **40-pin chip at U7.** If the part number stamped on it reads **"CO21697"**, you are lucky! You may use the simpler circuit. If the number reads **"CO12296"**, you must use the larger circuit.

Assemble the appropriate circuit on a two by three inch circuit board (Radio Shack's **#276-150** is ideal). Use very low profile sockets or no sockets at all, as height is severely limited by the shielding. If you use no sockets, be careful not to apply heat to the IC pins for too long a time. Keep the wiring on the chip side of the board to conserve space. The wiring must be soldered, as there is no room for wire-wrap posts. The board plugs into the socket at **U27** via a **16-pin DIP** header and short ribbon cable. Finally, install the jumper wires.

Find a resistor marked **R32** immediately behind the row of RAM chips and remove it. A trace from one of the holes runs to **pin 1** of the RAMs. Solder the first jumper to that hole. The next jumpers run to a parallel port which the Atari uses to control ROM switching.

We need pins **12 through 16**, which are normally unused and not connected to any traces. Locate **U23** and carefully pry the 40-pin chip from its socket. Bend up pins **12, 13, 14, 15** and **16** so that they point straight out. Reinsert the chip.

Cut five adjacent pin positions from an IC socket and solder the jumpers to them. Use this custom socket to connect the jumpers to the three protruding pins. Cover the connector with electrical tape, as the shielding is very low at this point. If you are using the circuit for the older 800XL, you must install an additional jumper. Locate a trace on the motherboard from pin **8** of the video controller, **U7.** Along the trace find a hole and solder the jumper there. Finally, insert a thin piece of stiff cardboard or plastic

# 256 UPGRADE FOR THE 800XL continued

under the small circuit board to avoid shorting the circuit. Refasten the shielding to the motherboard. If it doesn't fit over the circuit, carefully pound a dent out of the shielding with a hammer. Reassemble the computer. If all has gone well, the computer should power up and perform normally, although with the 256K in your XL, be sure to wait at least ten seconds after turning the computer off, else it may not coldstart properly when you turn it back on. The computer is ready to try some software that utilizes the large RAM space.

## The RAM-disk Software

Bank-select RAM is useless without software to control it. The software must obey strict rules as outlined above to work properly. The software must also be tailored to fit the application. Applications vary. For example, many graphics screens may be stored in the RAM, possibly to be displayed in quick succession for animation. Alternately, the RAM may act as a print spooler. A word processor would print an entire document quickly into the RAM and go on to other jobs while the RAM empties slowly to a printer. These applications are rather specific and might not appeal to all users.

A more universal application is the RAM-disk, a RAM-based disk drive simulator. To DOS and to the user's programs, the RAM-disk appears just as another disk drive, except that it is very fast. The application program may then use standard DOS commands to access the large RAM space. The 192K bytes of available RAM hold more data than two Atari 810 drives or one double-density drive.

The RAMdisk software I have prepared, **QMEGXLD.SRC**, offers a choice of either two single-density RAMdisks or one double-density. Also available is **QMEGXLS.SRC**, a RAM-disk program that sets up one single-density RAMdisk and leaves the XE-equivalent banks free for XE software. This is quite useful with **BASIC XE**, **DOS 2.5**, or the new **Synapse** software.

Assemble the source code with any assembler that accepts the syntax of the Atari Assembler/Editor. Assembly produces an object file that performs several tasks as it loads. First, it copies the operating system from ROM into the underlying RAM. Next, the RAM-disk routines load into the RAM-based OS, overwriting the international character set, a little-used feature of the 800XL. Lastly, it patches the OS to install the RAM-disk program and calls DOS's initialization routine to let DOS recognize the new drive.

The source code allows two options: the drive number and the density.

## 256 UPGRADE FOR THE 800XL continued

The RAM-disk can act as any drive numbered 1 to 8. If you have one real drive, you might want the RAM-disk to be drive number 2. Remember that your DOS must be set up to look for the drive number chosen. See the DOS manual for instructions concerning drive numbers. Typicallly, you must POKE memory location 1802 with a value of 15 in order to recognize disk drive numbers up to #4. Then you will write new DOS files which will thereafter always include your POKE.

The RAM-disk object file should be made to boot in after DOS so the user needn't worry about it. In Atari DOS, naming the file **AUTORUN.SYS** accomplishes this. Once the object file has loaded, the RAM-disk **MUST BE FORMATTED** before use. You may do it manually from DOS, or the application program may do it automatically (use the **BASIC XIO** command or a call to **CIO** in machine code). **RESET** won't harm the contents of the RAM-disk, nor will rebooting the computer, as long as the computer is not turned off (to reboot without powering down, **POKE 580,1** and press **RESET**, or jump to **$E477** in machine code). After rebooting, the RAM-disk program must be reloaded to access the data, which should then be found unharmed in the RAM-disk. This is why the RAM-disk program does not automatically format the RAM-disk upon loading. The major disadvantage to the RAM-disk approach is that all data is lost when the computer is turned off. The application must take care to save important data to a real disk before ending. However, the speed, convenience, and versatility of the RAM-disk overshadow its drawbacks.

### Uses

An assembly language programmer, after studying the RAM-disk source code and heeding the rules above, can devise many practical uses for a quarter-megabyte of RAM. The large RAM space, joined with the Atari's versatile hardware and low price, gives a performance/price ratio that is unbeatable in today's microcomputer market.

☞ **I ask one thing in return for this information: Please pass it around to all your interested friends. Put it in your club's library or on your favorite BBS. Encouraging software support of 256K will result in many interesting uses for it. Thank you and enjoy!**

DEFINITION OF MEMORY CONTROL REGISTER AT $D301 (54017 DECIMAL)

☆ XL MOD bit: 7 6 5 4 3 2 1 0 D a b E c d B R D=0 enables diagnostic ROM

☆ B=0 enables BASIC ROM

# 256 UPGRADE FOR THE 800XL continued

- R=1 enables OS ROM
- E=0 enables extended RAM
- abcd is 4-bit extended RAM
- bank # - ranges from 4 to 15
- banks 12 to 15 are equivalent to XE's banks 0 to 3

## 130XE bit: 7 6 5 4 3 2 1 0 D
## V C x y B R

- D=0 enables diagnostic ROM
- B=0 enables BASIC ROM
- R=1 enables OS ROM
- V=0 enables extended RAM for video
- C=0 enables extended RAM for CPU
- xy is 2-bit extended RAM
- bank # - ranges from 0 to 3

## PARTS LIST

- 8 41256 256K-bit dynamic RAM (200ns or less)
- 1 74LS153 Dual 4-to-1 multiplexer (IC2)
- 1 74LS139 Dual 2-to-4 decoder (IC3)
- 1 - 33 ohm, 1/4 watt resistor

### ADDITIONAL PARTS FOR ANTIC #CO12296

- 1 74LS158 Quad inverting 2-to-1 multiplexer (IC4)
- 1 74LS393 Dual 4-bit counter (IC5)

### LIST OF CONNECTIONS FOR THE UPGRADE CIRCUIT

Instead of a drawing of the upgrade circuit, below is a list of connections. Each entry in the list begins with the name of the signal followed by all the IC pins that connect together and share the signal.

- **IC3-13 means pin 13 of IC3.**

The IC numbers appear in the parts list above. IC1 is the 74LS158 chip from socket U27 on the XL motherboard. DIP is the DIP header to be plugged into socket U27. U23-xx refers to the pins you bend up on the PIA chip at U23 on the motherboard.

- Vcc : DIP-16, IC1-16, IC2-16, IC3-16, IC3-13
- Vss : DIP-8, IC1-8, IC2-8, IC2-1, IC2-15, IC3-8
- A7 : DIP-11, IC2-10, IC2-11
- A15 : DIP-10, IC2-13, IC3-3
- A6 : DIP-14, IC2-6, IC2-5
- A14 : DIP-13, IC2-3, IC3-2
- MUX : DIP-1, IC1-1, IC2-2
- A4 : DIP-2, IC1-2
- A12 : DIP-3, IC1-3
- RA4 : IC1-4, DIP-4
- A5 : DIP-5, IC1-5
- A13 : DIP-6, IC1-6

## 256 UPGRADE FOR THE 800XL continued

☆ RA5 : IC1-7, DIP-7

☆ -E : DIP-15, IC1-15

☆ RA7 : IC2-9, DIP-9

☆ RA6 : IC2-7, DIP-12

☆ PB2 : U23-12, IC2-4

☆ PB3 : U23-13, IC2-12

☆ PB4 : U23-14, IC3-1

☆ PB5 : U23-15, IC1-10

☆ PB6 : U23-16, IC1-11

☆ -Zd : IC1-9, IC3-14

☆ -O1a : IC3-5, IC3-15, IC2-14

☆ -O2b : IC3-10, One side of resistor

☆ RA8 : Other side of resistor, Pin 1 of all RAMs

If your **U7** part number is **CO12296**, do not connect signal **A7** above, and make the following additional connections.

The connection to **U7** is to a trace on the motherboard that runs from pin 8 of U7.

☆ Vcc : DIP-16, IC4-16, IC5-14, IC4-3

☆ Vss : DIP-8, IC4-8, IC4-2, IC4-15, IC5-7, IC5-2, IC5-12

☆ A7 : DIP-11, IC4-6

☆ -REF : U7-8, IC4-1

☆ REF : IC4-4, IC5-1

☆ A7' : IC4-7, IC2-10, IC2-11

☆ Q7 : IC5-8, IC4-5

☆ Q3 : IC5-6, IC5-13

## Any Problems?

Did you check your issue 19 disk as you received it, back in January?

We had a phone call saying that there are about three programs that wont run, they come up with an error message. We checked the master disk and found it to be alright, but when checking some of the copied disks we found that they were indeed faulty.

To save all of you complaining we have included with this issue on the B side of the PD Library disk the programs from the B side of issue 19.

You probably wonder why we have included a PD Library disk, the reason is to save us re-printing an updated PD list every two month. We will put the updated list on a disk or include it on the issue disk and inform you accordingly. The PD file can than be transferred to the main PD disk.

So KEEP the PD disk safe.

We hope this will be satisfactory.

# REVIEWS

**By Kevin Cooke**

**Hello again. If this article doesn't appear in issue #19 of TWAUG's magazine, I apologise - with my commitments to college work, I've been snowed under lately!**

However, I think you'll enjoy this review column, it looks like being another long one. Without further ado, here goes!

## Title: TECHNUS

Sold by: Micro Discount, 265 Chester Road, Streetly, West Midlands B74 3EA, Tel: 0121 353 5730

Price: 4 Pounds (+ P&P)

Technus is one of the latest additions to Derek Fern's range of software. The game first starts off with a picture which appears showing a man running along a platform. A scrolling message runs underneath (in Polish) and a good musical tune also plays along. A press of the fire button loads the main game.

The main title screen gives quite a shock when it first appears. Not that it is at all graphically exciting but an EXCELLENT digitised tune is playing with a backing of normal computer music - this is one of the best examples of digitised music which I've ever had the fortune to hear.

Again, pressing the fire button brings about the main game screen. The first thing that always strikes you in a game is the graphics. Technus' graphics are of a good quality - the character which you control looks good and is quite large whilst the platforms, etc. are all quite clear.

At the top of the screen is a beating heart (again, graphically very good) and below this is your character's energy bar. As the energy bar decreases, the heart starts to beat faster - a nice touch.

I have no instructions for this game so I cannot say what the object of it is. Basically, it seems to be your job to move the character through the series of screens and kill the robots that lurk there.

There are basically two types of robot - those that cannot be destroyed and those that can. A quick press of your fire button whilst playing will make your character shoot one of his (limited) bullets and should this hit one of the destructible robots, it will be killed.

Sometimes, these robots will leave a bonus behind. So far, I've only encountered energy replenishing bonuses and extra bullet bonuses but it could be that there are other bonuses which need to be collected.

To get from platform to platform and so discover new rooms, it is obviously necessary for your character to jump. Luckily, the character can still be controlled whilst he is in mid air so not only can you change his jumping direction mid-flight but you can also control him as he's falling from a platform - very handy!

The in-game music is good - not up to the standard of the incredible digitised music on the title screen but good nonetheless. If you prefer, the music can be dispensed with in favour of suitable sound effects.

Overall, another good game to add to your possible-purchase list. It's good to see that it's at such a reasonable price as well - support Derek Fern for supporting you.

**Title: STARBALL**

Sold at: Micro Discount (see above for address)

Price: 4 Pounds (+ P&P)

If you're looking for something different from the run-of-the-mill platform game or shoot-'em-up, why not take a look at Starball?

The game starts off with a Polish title screen and a lively piece of music playing in the background. A press of the fire button brings about the main game screen.

The object of Starball appears to be to guide a small ball around the play area and collect all of the diamonds which are laying about to progress to the next level. These diamonds also replenish the on-screen timer which quickly counts down.

Sounds easy? The game is made more difficult by the walls, locked doors and all manner of traps which lay about the scrolling level. Locked doors require keys to open them yet these keys often involve careful maneuvering to be picked up in the first place!

In case you haven't already guessed, the game has a certain puzzle element which makes it all the more fun!

The game itself is VERY addictive. The balance between careful movement of your ball and the puzzle of gaining each diamond makes the game a joy to play.

And good use has been made of the available colours which makes the game-screen very clear yet colourful.

The fun music which plays in the background throughout is rare in that it doesn't seem to get annoying like most tunes due to it's constantly changing nature!

The only bad point which I can find

with this game is that the on-screen messages are in Polish. Apart from that, I don't know how else the game could be improved.

Starball is one of those rare games that will appeal to both puzzle addicts and those that like more action-packed & fast moving games. I didn't find it particularly easy but at least the author provided a number of lives to play around with!

I have no hesitation in recommending this game - I thought it was brilliant! At four pounds, it's even more of a bargain.

**Title: TRON**

Sold at: Micro Discount (see above for address)

Price: 4 Pounds (+ P&P)

If you've played one of the PD Tron games, you'll know what to expect from this commercial release.

The object of Tron is to guide your futuristic bike around a screen at the same time as another player or computer does the same with their bike. Each bike leaves behind a "trail" which will destroy any bike that crashes into it. The object of the game is to make your opposing bike crash, ie. by cutting them off and making them crash into either one of the trails or the wall of the arena.

The title screen appears soon after loading. From here, you can select either a one player game (using a joystick in either port #1 or #2 - your choice) or a two player game.

A scrolling message travels along the bottom of the screen (unfortunately, in Polish) and some interesting music plays in the background.

By pressing start, you can play the game itself. In the bottom half of the screen are two windows - one for player 1's view and one for player two's view. Here, you can see your futuristic bike in detail and a small portion of the bike's trail which may be immediately around it. The scrolling effect here is created by the grid which lies below your bike -in actual fact, your bike stays central and the grid moves.

In the top half of the screen is another window showing the entire arean and the trails which the bikes have left behind, as in an ordinary Tron game.

To either side of this window is the score of each player and the number of lives which they both have left. These lives are represented by pictures of the player's bikes - the bikes which are smashed-up are no longer available. This is a nice touch.

The only way in which you can travel through one of the bike's "trails" is to press the fire button on your joystick. This make's your bike flash for a short period of time in which you are indestructable - this function is limited to only three uses per. life, however.

The graphics are all created in different shades of blue. In other games, this type of colourscheme could be considered a bit of a let-down but luckily it seems to suffice for this type of game. The graphics are nothing stunning but, again, are fine for this type of game.

Whilst you are actually playing, the only sound is that of your bike's motor. A short musical piece plays after each of your eight lives expire.

With regards to gameplay, Tron has little more to offer than it's PD counterparts. However, the game screen with the two different views is unique and adds an extra dimension to gameplay that keeps you wanting that extra go.

Tron has always been addictive (even if you've never wanted to admit it!). If you're looking for the definitive version of Tron or a decent simultaneous two-player game, take a look at this!

**Title: WHERE DINOSAURS RULE**

Sold by: J.F. Software, 26 Auckland Road Kingston Upon Thames, Surrey, KT1 3BG,

Price: 5 Pounds (inc. P&P) (cheques payable to "J.Foskett")

Where Dinosaurs Rule is the latest text adventure from J.F.Software (a.k.a John Foskett).

The story which accompanies the adventure says that you have found a land, accessible only through a network of caves, where dinosaurs still rule. The authorities apparently know about it but you feel that you should prove to the world that it exists.

So, into the caves you go.... If you read my review of Rose Gardens in issue 17, you'll know what to expect from Where Dinosaurs Rule.

A standard graphics 0 text screen is split into a number of different sections and each of these have been given different colours for easy recognition - a nifty piece of programming! This dividing of the screen allows you to always have your character's inventory, location description, exits, etc. on screen.

The game's parser (understanding of words) is very good and it will normally understand what you are trying to say.

As I've already said, the game screen is colourful and so also attractive to look at. To be honest, there's not much you can say about an adventure game. The real test of this type of game is whether is is programmed well enough and whether is contains enough puzzles & atmosphere to keep you interested. On these points, Where Dinosaurs Rule scores highly and so can easily earn my recommendation. If you have even the slightest interest in adventure games, take a look at this - you won't regret it.

I'm sure that John Foskett would like to thank those readers who sent for his last text adventure, Rose Gardens. Well, actually NOT ONE READER DECIDED TO SEND THE 1 POUND 50p for it. If this is the sort of response that Atari supporters get then it's no wonder that so many companies are giving up and moving on to the PC. Fair enough, text adventures may not be your thing BUT why not send just this 1 pound to show your support?

John doesn't just produce text adventures - he also has a range of high quality utilities for sale but without any support, he won't continue. If you don't want any new software or just can't afford it at the moment, why not send a letter to tell him so? I'm sure he would rather be told that his software is not wanted or that you can't afford it at the moment rather than receive no response at all. So write that letter today or, better still, send for WHERE DINOSAURS RULE or ROSE GARDENS and show your support before there is no one left to support.

That's enough of my preaching. Now do the right thing.

[I would like to dedicate this article to the memory of Dave Ewens. I'm sure everyone who knew him will agree that he will be missed more than words can express.]

## FOR SALE

Various Atari hardware including:

Upgraded 130XE, 1050 Disk Drives with various upgrades i.e.:

U.S.Doubler, Happy, Laser, Quad Board.

Other items also available, including Touch Tablet, Light Gun & Modem.

Ring John Tel. 0191-262 6897

Armstrad PCW 9512 Personal Computer Word Processor including Printer, in perfect working order. It is an ideal computer when you are doing a lot of writing. Manual and original plus spare disks included. Also a Modem.

Contact Max for more details and price.

Tel. 0191-586 6795

## FOR SALE

10 pieces of Hardware, including:

A standard 130XE and 65XE.

Two printers 1020 & 1027.

Touch Tablet and Light Gun.

1050 Disk Drive & Superprofessional Joystick.

XC12 Datacorder, 850 Interface

All Leads, PSU's & Connectors.

The computers, disk drive and joystick have little used, the rest of the hardware have not been used at all.

Loads of Software including:
Atariwriter Plus Cartridge, Mastertype Cartridge, Rescue on Fractulus Cartridge, Crossbow Cartridge, Pacman, Silent Butler,Nibbler Disk, Movie Maker, Pole Position, Timewise Disk, Atari Logo Cartridge, Visicalc, The Bookkeeper and Keypad, Basic Compiler disk, Superscript, The Home Accountant, Mini Office II, Invitation to Programming 2 & 3, Assorted Games Demos etc. Disks and Cassettes.

Word processors and demos have been used, but all the rest is unopened.

Original Books and Manuals.

Contact: Mr.Brian J.Taylor, 24c
Castleton Court, Castleton Drive,
Newton Mearns, Glasgow G77 5LW.
Phone/Fax 0141-639 6516

## XL TWO

In Issue 11 of the TWAUG newsletter, I wrote a bit about XL-2. I (MAX) explained how to install the program and also how to set the clock. TWAUG was asked by John Picken, the author of this versatile program, at the beginning of last year, to test it thouroughly. It was after I used it a while that I came across a number of bugs, including losing a lot of time, when set, in a day.

Well, we have received the latest updated and modified version 3. It is a double density two disk set of XL2 and one disk of MYDOS 4.51XE. Both sides of the XL2 disks are in ARCed format, the MYDOS disk is in ARCed format on side B only. It is also ideal for programmers as all source code files are on the disks.

For those readers who haven't seen September/October 94 issue here is short re-cap. XL2 is a program for the XE or 800XL that gives the user two screens to work with. Once installed, you have two identical computers in memory and you can run a different application in each. You can switch between the system by just pressing CONTROL-HELP. It is actually better than using the RamDISK, when you want to use another program from the RD you must re-run it. With XL2 there is no need to re-run the programs they are intact and the switching only takes about two seconds. The XL2 program is

## XL-TWO cont

not multitasking, only one system is in operation at any one time, the other system is on hold.

**DOS SYSTEM**

SpartaDOS 3.2 or 2.3 can be used as well as BW-DOS, DOSXL or MYDOS. With SpartaDOS 3.2, a saving of 600+ bytes at low memory using XL2 in place of TDLINE. With XL2, low memory usage under SpartaDOS 3.2 is absolutely zero. I must mention here that XL2 is incompatible with SpartaDOS X.

I mentioned all the good points in issue 11, I only want to mention that the author made vast improvements and rectified the bugs in the program.

I had many queries about the way I run Textpro and Daisy-Dot from memory, I've got a 1 meg upgrade. If you only have a standard 130XE, well here is the answer, one screen can hold Textpro and the other Daisy-Dot. You need an upgraded 800XL to run XL2, but the modified MYDOS 4.51 provides a small RamDISK on an 800XL.

By John Picken

October 1995.

This XL-2 three disk set is now available from our library. { ED.}

## LETTER SECTION

24/1/96

A small note from Allan Doyley

Dear John & Max,

Just received issue 19 & am impressed by clarity of new format. Hope you can cope with all the extra work, with David's passing, which saddened me.

My wife's illness leaves little time for writing articles, but I do enjoy reading the Newsletter and trying out its contents.

Very best wishes,

Allan

Thank you very much for your note, your acknowledgement is appreciated. We are very sorry to hear about your wife's illness,

hoping she will get better soon.

Max

# DISK CONTENT

Side A of this issue disk contains a database program, it sorts your data and can be used as a mailing list. This DMS Database program allows you to add a dictionary which never comes in wrong. This is a program worth adding to your library.

There is Superplot, a Graphics program for those who like to draw pictures on the screen. This program is also a good way to learn the Y and X positions of the screen.

There are four games also. We'll start with Guardian, this is a Basic program and the aim is to protect the Humanoids from an Alien attack. You've got three guns to defend the earth with. You control the guns with the joystick plugged into port one. Pushing the joystick to the left you control the left gun, pushing forward for the centre gun and push to the right to control the right hand side gun. It isn't as easy as it sounds, you game players out there maybe luckier than I've been, result for my try is nil.

Hoppo, another Basic sort of board game. You must guide Hoppo, avoiding the alien attacking you, to pick up the spicky object that keeps moving about just when you think you've got it. I didn't get very far with this game either and so I am unable to give any more details on it. I know one thing though, you collect extra lives after you have been successful collecting a certain number of those spicky objects.

Termite, this is a fast moving game. Again use the joystick to control some sort of bug, that's what it looks to me, and the aim is to avoid the attacking Termite, which by the way moves very fast. You can move left, right, up down and pick up the objects lying about and there are plenty of those, if you can that is, the Termite is constantly attacking you.

On side B of the disk you'll find a nice little program SIGN MAKER. It is in Turbo Basic, it's all set up. You also find the documentation on the same side of the disk. This program lets you create and print files, and as the name says you can make large signs, posters and even include icons to include on birthday cards.

# 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**
**41 Henryson Road**
**Crofton Park**
**London SE4 1HL**

## 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.A.Pignato**
**O.H.A.U.G.**
**3376 Ocean Harbor Drive**
**Oceanside, N.Y. 11572**
**USA**

# TWAUG NEWSLETTER

## ADVERTISEMENT

## CHAOS! COMPUTERS

PO BOX 30
MANCHESTER M19 2DX
Telephone: (0161) 737 1946

### THE HYPER DRIVE

Upgrade your *ATARI 1050* disk drive with a **HYPER DRIVE** enhancement from **CHAOS! COMPUTERS**.

The **HYPER DRIVE** is an easily installed hardware & software package for the *ATARI 1050* which will enable your disk drive to back-up most disks protected by unreadable or badly formatted sectors. Most copied disks can then be loaded on any 1050, whether enhanced with a **HYPER DRIVE** or not. The **HYPER DRIVE** enhancement also offers fast reading, writing, formatting and copying in single, medium or true double density formats (i.e. it is compatible with *RANA*, *PERCOM* and *INDUS* double density drives, and will read *U.S. DOUBLER* type format).

Fitting the **HYPER DRIVE** coudn't be simpler and requires no special tools or soldering. It simply plugs into socket on the 1050 circuit board. And with our VERSION II software package and full 28 page manual, it is one of the most versatile disk drive enhancements /copiers you can buy. **HYPER DRIVEs** are available exclusively from **CHAOS! COMPUTERS** at a special intro-ductory price of just £30.00 each.

Please make Cheques/Postal Orders payable

to '**P. HOLLINS**'.

Prices are subject to change, from time to time, due to component costs, so wherever possible please *'phone to check*

Barbarian
ATARI
XL/XE
LK AVALON