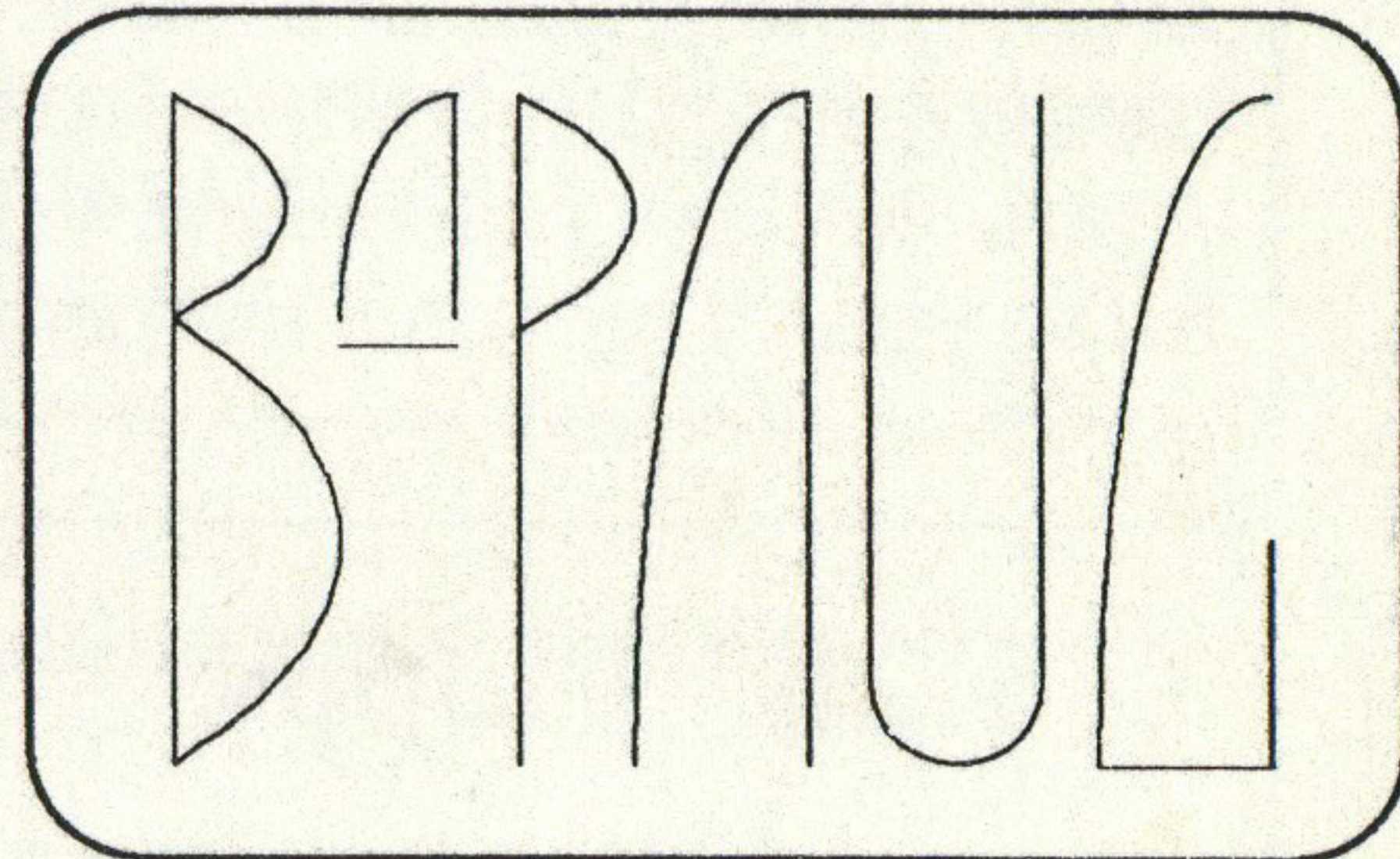
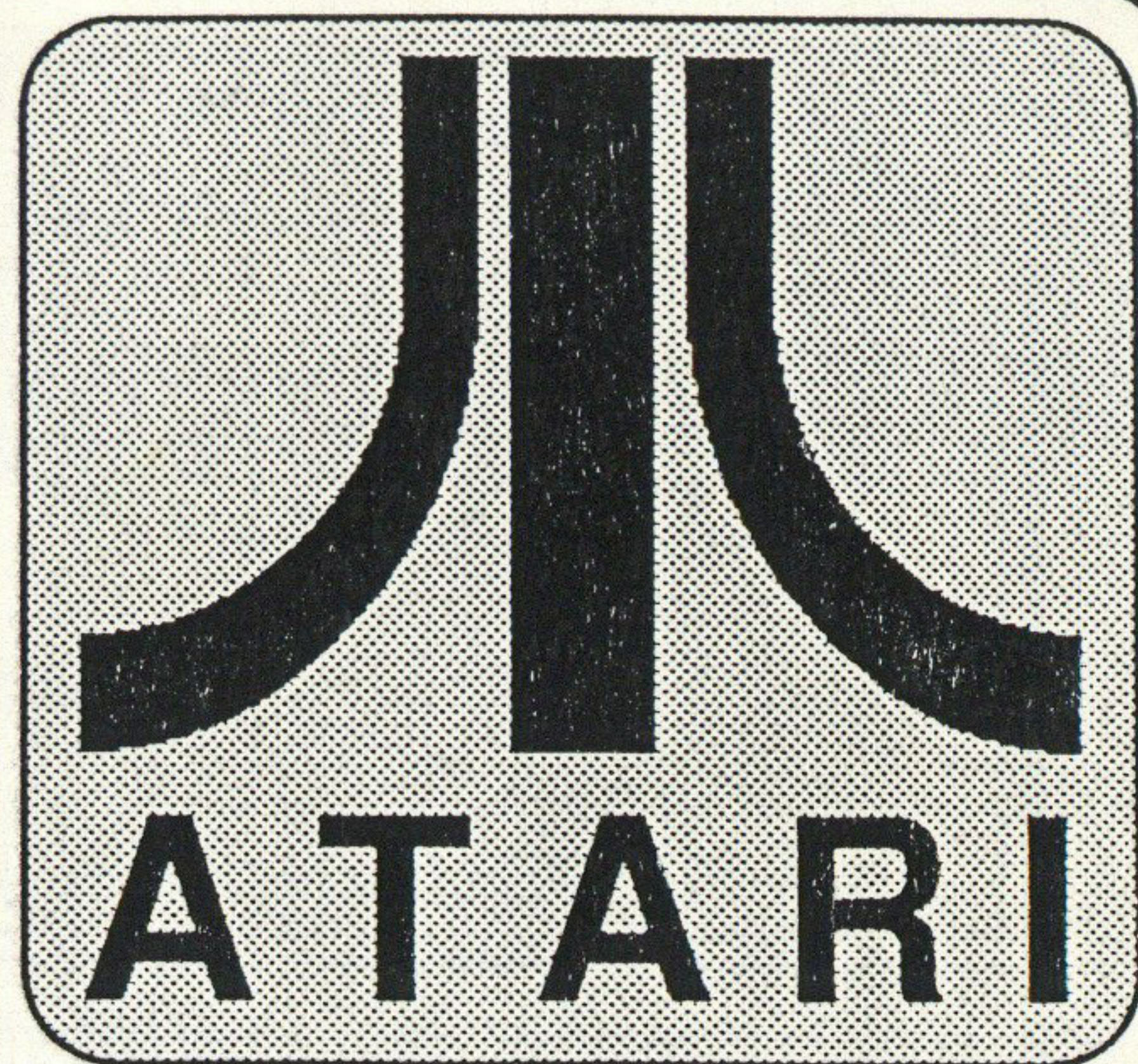




The Alternative Atari Newsletter



95p March/April 1990 Issue 9

Writing A Bulletin Board On The Atari 8 Bit

Assembler Programming Techniques for the Atari ST

Music Scene: MIDI



The 1989 BaPAUG Awards

BRUEN ENTERPRISES LTD.

47 Mandeville Road, Southgate, London N14 7NJ

Telephone: 01-368 4155

Bruen Enterprises is a new company supporting the 8 bit Atari. We are importing quality hardware and software from the states. We will provide any product or service for the 8 bit Atari. If there is anything you want for your 8 bit then please telephone us and we will try to get it for you. This advertisement details some of the products we can already supply.

5.25" Floppy

Discs

£3.50 for 10 £16 for 50
£30 for 100

Cables

850 to printer

850 to modem

2 metres long & only £8 each

Memory Upgrades

We will upgrade your computer for £60. XLs to 256K and XEs to 320K.

We can also supply 320K XEs for £120.

PD Software

* Cassette & disk based *

General

We buy, sell and part exchange all 8 bit Atari hardware. We have 1050 disk drives and 850 interfaces in stock. We can also supply parts and do repairs.

Cheques and postal orders should be made payable to Bruen Enterprises Ltd.

The Black Box

Only £200 for so much. It gives your 8 bit a printer spooler, RS232 interface with built in handler that takes no user RAM, hard drive interface and a button that dumps the contents of the screen to your printer when it is pressed.

There is also an add on that allows you to connect up to 4 standard high capacity drive mechanisms to your Black Box. The 4 drives are super archiver compatible with a capacity of upto 720K per disk. There is also software that allows the Black Box floppy interface to read and write to IBM compatible disks. You can have 3.5" or 5.25" floppies connected.

The Black Box needs no hardware modification. It just plugs into the parallel bus on your XL/XE.

Black Box and 20M Hard Drive

For £500 we can supply a 20M byte hard drive system including the Black Box (see above for details).

80M Hard Drive

* Please telephone for details *

Turbo 816

£120 for a 16 bit replacement CPU for the Atari 8 bit. It works with all existing hardware and software. It provides new instructions and addressing modes and allows you to connect upto 8MBytes of directly addressable memory to your XL/XE (not bank switched). It is great for machine code programmers.

XF551 Drive Modification

We can supply a drive modification for the XF551 that allows you to connect a 5.25" or 3.5" 720K drive mechanism.

* Please telephone for details *

Editorial Team

Colin Hunt D&E Controller
Thomas Holzer VCS Controller
Graham Broomfield Graphics Controller

All contributors are credited with their articles entry within the contents list.

8:16 is produced by BaPAUG Publishing.

All original articles published within 8:16 are the copyright of the author, but can be used by other publications provided the author and 8:16 are acknowledged as the original source and a copy of the publication is sent to 8:16. Articles re-printed from other newsletters should be credited to the original source.

8:16 and the BaPAUG cannot be held liable for any errors or claims made by advertisers.

Letters and articles should be sent to:

8:16
c/o 248 Wimborne Road
Oakdale, Poole, Dorset BH15 3EF

This issue of 8:16 was produced using a Atari 130XE, 1050 disk drive, 520STFM, SM124 monitor, Cumana 1M external drive, Star LC24-10, Apple Macintosh SE and Laserwriter Plus. The software used was TariTalk on the 8 bit. Timeworks Desktop Publisher, 1st Word Plus, Gem Kermit and TariTalk on the ST. Microsoft Word, MacPaint, MacDraw, Mac Kermit and Aldus Pagemaker Version 3.5 on the Macintosh. One day this will be an Atari only publication - I promise.

The opinions expressed within 8:16 are those of the authors and are not necessarily held by the BaPAUG.

The BaPAUG is a non profit making organisation.

This issue of 8:16 is dedicated to the people of East Germany and to the destruction of the Berlin wall. This issue is also dedicated to Orbis Records for supplying me with many hours of Thin Lizzy on CD.

Copy date for the next issue of 8:16
is 5th April 1990.
Issue date is 4th May 1990

Glen expresses his regret that there is no SIGnews section due to the pressure of other work.

Contents

Page 3

8:16

Issue 9

Features

- 8 Bit Reviews5
Short reviews of the latest games available from B. Byte.
By Ian Brooker.
- Assembler Programming Techniques
For The Atari ST6
Within this issue we examine methods of
generating random numbers.
By Peter Hibbs.
- Stateside 8 Bit Products8
A list of products available within the states.
By Colin Hunt
- The Xtra-RAM Upgrade Board10
Review of this useful ST upgrade.
By Graham Jones.
- Introduction To C Programming12
The second article in our C series covers Arithmetic and
Logical Operators and Functions.
By David Watson.
- 8 Bit Review: Speed Run15
Reviewed by Ian Brooker.
- Music Scene: MIDI16
A new series of article covering the use of MIDI and MIDI
related software. This first article reviews Super
Conductor.
By Paul Brookes.
- Using DOS 2.518
Second of 4 articles on using DOS 2.5. This issue we cover
MEM.SAV, RAMdisks and the menu options relating to
binary files
By Colin Hunt.
- The 1989 BaPAUG Awards22
- 8 Bit Review: Mad Jax23
Reviewed by Ian Brooker.
- Writing A Bulletin Board On The Atari 8 Bit ..24
Everything you wanted to know about producing a bulletin
board, complete with detailed instructions on how to
program the Atari 850 interface.
By James Bastable.

Departments

- Notice Board4
Bits, Bytes & Words7
BaPAUG News11
8 Bit Software Roundup22
User Group File30

**The Bournemouth and Poole Atari User Group is a member of
The Association of Atari User Groups.**

Notice Board

New ST Hard Drive Supplier

We have just received details of a new supplier of DIY and complete hard drives for the Atari ST (along with others) range of computers. Discdrive Ltd can supply NEC 3.5" half height, NEC 5.25" full height, Seagate 3.5" half height and Seagate 5.25" half height drives with SCSI interfaces. The price for the Seagate ST 125N (20MB) is £290.

Two ready built hard drives are available; the DDL ATH20MG with real time clock, DMA and SCSI ports, SCSI and ATARI mode switch and aux mains output socket for £399. A 48MB version is available for £499.

Ready built boxed floppy drives for the ST include a 3.5" 720K external floppy with integral power supply for £75, a version powered via the mouse port for £60 and a 5.25" 720/360K external drive for £120.

For more details contact:

Discdrive Ltd., Unit 87, Ellingham Industrial Estate, Ashford, Kent TN23 2NF

Telephone: 0233 610990 (sales), 0233 646580 (technical).

Atari UK Sell Off All 8bit Hardware

Atari UK have sold all their 8 bit hardware to Derek Fern, of the Birmingham User Group (BUG). Amongst the hardware there were returned 1050 drives, 1020 printer plotters, 1027 printers and tracker balls. For more details Derek can be contacted on 021 353 5730.

8:16 Subscription Rates Annual (4 issues)

U.K.	£3.80
Europe	£8.00
Elsewhere (sea)	£8.00
Elsewhere (air)	£15.00

8:16 Commercial Advertisement Rates

Full page	£40.00
Half page	£20.00
Quarter page	£10.00

Phone (0202) 677895 for details.

Atari Portfolio User Group

Those of you who have purchased the excellent Atari Portfolio will be interested to hear that there is now an independent user group aimed at novice and expert, business and hobbyist Portfolio users.

The group produces a monthly magazine (newsletter) that includes news on new items, reviews of hardware and software, information, tips and advice.

If you would like to join the user group, which cost £18 a year or £10 for six months, an application form can be obtained from:

Atari Portfolio User Group,
84 Cambridge Avenue, Gidea Park,
Romford, Essex RM2 6QU
Telephone: (0708) 730764

Bruen Enterprises

It is always good news to see new companies supporting the Atari 8 bit. Bruen Enterprises are one such company who are specializing in the import of hardware and software from the US.

Please note that the telephone number on their advertisement (page 2) is no longer valid. The new number is **0831 166429**.

New City

James Bastable has re-written the City BBS software. If you are new to comms, we strongly recommend that you take a look at this BBS, just to see what is possible with the Atari 8 bit.

Diamond GOS

The Diamond GOS cartridge is now available from **Frontier Software** for £49.99. Frontier do not guarantee to stock GOS, but will order it from the states for anyone who wants it.

Diamond GOS is the ST GEM lookalike user interface produced by Reeves Software.

Frontier Software can be contacted on **0423 567140**.

XF551 Drives

Computerhouse UK have available the last UK compatible XF551 disk drives for £165. Stock is very limited and these drives will not be available again.

Computerhouse, 14 Romily Court, Landridge Road, Fulham, London SW6 4LL
Telephone: 01-731-1276

8-BIT REVIEWS

There is yet another software house fresh on the scene producing Atari XL/XE games. B. Ware Software have just released 4-titles for your favourite 8-bit. These are **Zoltan Escape**, **Cybor-Stien**, **Escaper** and **DARG**. These are, at present, only available on disc and the former three are reviewed here. Future titles available are **The Gorblasters**, **Invasion of the Zip Heads**, **The Mutoids** and **Alien Brain Eaters**.

Escaper

The Great Zorgoid (magic ruler of the Universe) has decided to destroy earth because of all the non Atari computers it produces (and why not). As an Atarian (the only Humanitarian amongst computer users) you have been given the chance to redeem your fellow men and save the day.

In the lost city of Atlantis you must open the code clocks, automatic doors and bridges and fight your way to the trans dimensional ejector tube and escape. The city, as with the castle in **Cybor-Stien**, is well drawn, but this time with better sound effects. The playability is let down by a swarm of flys or something that follow you wherever you go sapping your energy and whenever you stay still too long. You must therefore keep going with no time to rest or think as you encounter each new screen. One word of advice, in order to complete some screens you need to perform a task on the next, so play ahead. On the whole a nice game.

Machine: 48K
Cost: £5.99
Sound: 5
Graphics: 8
Playability: 6
Value: 6

Zoltan Escape

You have crashed on the planet Zoltan and with your space ship having no fuel for escape you must put on your jet pack and fly through a radioactive complex collecting uranium pods. Security beams come on and off across your path. Touching these beams, or the walls and floors means instant death, so you have a

tricky path ahead.

The graphics are reasonable and colourful, sound is limited to the beams and your jet pack, there being no background music to keep your mind working while you fly. To operate the jet pack you press the fire button and then left or right to move sideways. Releasing the fire button brings you back down to earth (sorry Zoltan). To complete each level you will need to develop a very accurate trigger finger - to much boost and you hit your head, too little and your feet touch the ground. In both cases the result is instant death, therefore a good quality joystick is essential. To add to your troubles each level has a time limit, though to be honest at first you will find that this doesn't play a part, as you will be dying a lot quicker anyway.

This particular type of game does not suit my taste and coupled with the fact that you will need a lot of practice to get used to controlling the character, I found myself getting board very quickly - dying every other second is not fun. Maybe that's an excuse and the game is just to difficult for me, but either way it was just to much of a challenge. Die hard platform freaks will find this game a challenge, with the added twist in that you have to stay clear of the platforms in order to stay alive. For those who find the game to difficult it would of been nice if a 'cheat mode' was available where the captured pods stayed captured when you die.

Machine: XL/XE only
Cost: £5.99
Sound: 4
Graphics: 7
Playability: 3
Value: 5

By Ian Brooker

Cybor-Stien

You play the part of Gregor, the hunchback servant of a mad professor. The only problem being that your master, the professor has been captured by the villagers down the hill. To save him you must find all the parts of the mad professor's unfinished project, the **Cybor-Stien Monster** and put it together so that 'it' can rescue him. The only problem is that all the pieces are scattered throughout the professors dungeons. To find the pieces you have to avoid the ghosts, skeletons and ray beams, jump over obstacles and go up and down the platforms collecting the parts in this well drawn multi-screen castle dungeon.

As with **Zoltan Escape** the sound is limited but otherwise this is quite a nice game and fairly playable with each screen requiring careful planning. If you do find the game impossibly to complete the brief instructions do tell you about a cheat mode, though you will have to be a particularly small and weedy example of a Gregor to use it.

Machine: 48K 8-bit
Cost: £5.99
Sound: 5
Graphics: 8
Playability: 7
Value: 7

For more details on B. Byte software for the Atari 8-bit send a self addressed stamped envelope to B. Bytes, 19 Southfield Road, Hinkley, Leicester LE10 1UA.

Random Number Generators

by Peter Hibbs.

It is often necessary, especially in games programs, to generate a random number for use by the program. It is not easy, however, to generate a true random number without special hardware such as a noise generator or similar. Computer programs usually have to be content with pseudo-random generators which give the appearance of being random over a large number range and are generally good enough for most games applications. A pseudo-random number generator starts with a value in a store called the 'seed' and then performs some mathematical operations on it to produce a different number. This number is then stored back in the 'seed' in readiness for the next number. The ideal generator should produce all possible combinations of numbers before repeating itself. For example, a 16 bit seed should generate 65536 different numbers in succession. One advantage of a pseudo-random number generator is that if the value of the seed is known, the next number generated can be predicted. This can be useful for testing programs which generate random effects. If the seed is loaded with a specific value, the same sequence of random numbers will be generated each time the routine is called.

ATARI BIOS CALL 17, RANDOM.

The Atari ST operating system provides a built-in random number generator. This generates a 24 bit pseudo-random number in the range 0-16,777,215. The seed for this number is copied from the 200Hz

Generate 16 bit random number and store back in (seed)
ENTRY (seed) contains 16 bit value
EXIT d0 and (seed) contain new 16 bit value

```
random2  clr.l    d0                clear d0 (all 32 bits)
         move    seed,d0          fetch last seed
         mulu   #58653,d0        multiply by 58653
         add    #13849,d0        add 13849
         move   d0,seed          restore seed
         rts
seed     dc.w    1                16 bit store
```

Listing 2. 16 BIT PSEUDO-RANDOM NUMBER GENERATOR.

system clock on power up to provide a different seed each time the computer is used. This call is not a true pseudo-random number generator, however, because each time the seed becomes zero it is reprogrammed from the system clock and cannot, therefore, be predicted. A sub-routine to provide a random number from the BIOS is shown in listing 1. The seed store holds the new 32 bit number and is at address \$29B8.

16 BIT PSEUDO-RANDOM NUMBER

For most games a 16 bit random number is usually sufficient. Listing 2 shows a simple routine to generate a 16 bit pseudo-random number.

This routine simply takes the current seed value, multiplies it by the magic number 58653, adds 13849 and stores it back in

the store. For a more detailed explanation of this and the following routine the excellent book called 'Microprocessor Programming for Computer Hobbyists' by Neill Graham (TAB BOOKS) should be consulted.

If this routine is used some care should be taken over the starting value for the seed since the ST operating system usually clears all memory locations to zero before loading a program. This means that the seed would always start as zero and could give the same sequence each time. The seed should, therefore, be set to some random value at the start of the program. This could be done by a counter counting the delay between the first display screen and the user pressing a key or by copying the x and y co-ordinate values added together to the seed.

SCALED RANDOM NUMBERS

In some programs it is often required to generate a random number within certain limits. For example, in a card game to select a card at random, a number between 1 and 52 may be required. If the starting number is between 0 and 65535 (16 bit number), the output from the routine needs to be scaled down proportionately to a value of 1-52. Listing 3 shows a sub-routine which can provide a number within two predetermined values which are defined as in-line data.

ENTRY (seed) holds last number generated
EXIT d0=24 bit random number
(32 bit number with bits 24-31 reset)
(seed) holds new random number

```
random1  movem.l  d1-d2/a0-a2,-(sp)  save registers
         move    #17,-(sp)          random
         trap    #14                call bios
         addq.l  #2,sp              correct stack
         movem.l  (sp)+,d1-d2/a0-a2  restore registers
         rts                        return
```

Listing 1. BIOS RANDOM NUMBER.

Listing 3. SCALED PSEUDO-RANDOM NUMBER GENERATOR.

Return random number within specified range
 ENTRY inline data specify range (lowest+highest)
 EXIT d0.w=16 bit value in range

r_scale	movem.l	a0/d1-d4,-(sp)	save registers
	move.l	20(sp),a0	fetch return address
	move.l	(a0)+,d0	copy low limit to d0
	move.l	(a0)+,d1	copy high limit to d1
	move	d0,d2	copy d0 to d2
	sub	d0,d1	calculate difference
	addq	#1,d1	correct d1
	move	d1,d3	and copy to d3
	move	#17,d4	calculate number of bits
r_scale1	subq	#1,d4	dec d4
	lsl	#1,d3	shift high bit to Cy
	bcc	r_scale1	jump if bit=0
	bsr	random2	fetch 16 bit random No'
	lsl	d4,d0	shift right d4 times
	mulu	d1,d0	multiply by high limit
	neg	d4	
	add	#16,d4	calculate shift value
	lsl.l	d4,d0	shift right d4 times
	add	d2,d0	add difference
	move.l	a0,20(sp)	restore return address
	movem.l	(sp)+,a0/d1-d4	restore registers
	rts		return

The operation of the routine is described briefly, for more information consult the book mentioned above. The basic routine provides a value from zero to n-1 where n is the maximum value required. To obtain a value greater than zero the result is offset by the same amount. For example, to generate a value between 30 and 100, a value of 0 to 70 is returned by the routine and then 30 is added to it to give the required range. On entry to the routine the actual range is calculated by subtracting the lower number from the higher number to give a value between 0 and n. The significant number of binary bits is then calculated and stored in d4, i.e. if the value was 100 decimal, this is 1100100 in binary which is 7 significant bits. A 16 bit random number is then generated and shifted right by the number of significant bits in the original number, i.e. 7 if the value is 100 decimal. The result is then multiplied by the original number, i.e. 100 and the result of that is shifted right again by the 16's complement of the number of

significant bits, i.e. 9 is 16-7 where 16 is the number of bits in the random number and 7 is the number of significant bits in the required number. The original difference is then added to this result and returned in d0.

To call the routine the number range is defined as two 32 bit numbers immediately following the sub-routine call. For example, to generate a number in the range 27 to 515 the code would be as

follows:-

```
call routine
    bsr      r_scale
define range as longwords
dc.l      27,515
etc...
```

A number from 27 to 515 inclusive will be returned in register d0. No other registers are changed. Note that this routine calls the 16 bit random number generator described above.

Bits, Bytes and Words

This new regular section within 8:16 is intended to be a place where users can tell each other about unusual aspects of Atari Computing. Little bits of information that may be useful to other people. For this to work it requires input, so if you know something you think other may like to know then drop us a line. We will also use this column to inform readers of any changes to special offers and requests for new ones.

COPYMATE

Did you know that if you copy COPYMATE using copymate the sector map forms the digits of the version you are using. A neat check which must of taken sometime to implement.

Auto Booting GEM Applications

With the new version of TOS (1.04) it is now possible to autorun a GEM application. The information is stored within the DESKTOP.INF file via the following line:

MAC/65

Please note that we can no longer supply MAC/65 Cartridges at £39.95. We can still supply the cartridges but at time of going to press we are not sure of the price. Further deatails within the next issue of 8:16.

Stateside 8 Bit Products

With the complete lack of new Atari 8 bit hardware products and the low level of software being released within the UK, and Atari UK no longer shipping the XF551 disk drive the only possible place where anything new will be produced is the good old US of A. So when a colleague of mine came back with some 1989 Current Notes, an Atari Explorer magazine and Atari Interface magazine I dived in to find out what was available and from whom. Along with a recent issue of Antic, the following list was produced. Wherever possible I've given the companies telephone number, if this wasn't available I provided the address. Those products that are being imported by a British company I have stated the companies name and telephone number (if known).

Alpha Systems

0101 216 467 5665

Fancy trying some sound sampling, then the **Parrot II** (\$59.95) sound digitizer may suit your needs. A demo disk (\$5) is also available, along with a pre-recorded sound disk (\$4.95). On a different note **Pop-N-Rocker** (\$24.95) is a 'fast paced, multi-player trivia game that mixes questions with real songs digitized with Parrot', sounds interesting. If your main interest is with graphics AS produce the **Computereyes** and **Magniprint II Plus** as a complete digital portrait studio (\$119.95). Other products include **Chipmunk** (\$34.95), **Hack Back** (\$99.95), **Cheat** (\$24.95), **BASIC Turbocharger** (\$24.95), **Graphics Transformer** (\$22.95) and three books; **Your Atari Comes Alive** (\$24.95 including disk) and **Atari Software Protection Techniques Volume 1 & 2** (\$24.95 each or both for \$39.95 including disks).

American Techna-Vision

0101 415 352 3787

A supplier, who specialise in the supply of replacement chips (ANTIC, CPU, etc - \$4.50 each), bare PCBs for 400 and 800 (set prices from \$28.50), service repair manuals (from \$24.95), 1050 Tandem drive mechanisms (\$47.50) and a host of other goodies including a comprehensive software catalogue that includes titles not released here and titles I thought were deleted.

B & C Computer Visions

0101 408 749 1003

Supplier of a large selection of software on disk and cartridge, along with the odd piece of hardware. Among the software languages available there is **Lightspeed C** (\$35.95), **Draper Pascal** (\$44.95) and **Kyan Pascal** (\$62.95). B & C also have a large selection of education titles from companies such as CBS, Designware, Tink Tonk, Unicorn, Weekly Reader, Fisher Price and Spinnaker.

Bensley Consulting

P.O. Box 301, 217 West Walnut, Westfield, IL 62474.

A small family concern BC produce a range of Atari 8 bit programs under the name Happy Programs. Titles currently available are **Happy Programs - Astronauts** which is a quiz / learning tool for students and adults which aims to strengthen their knowledge of the space program, and **Happy Programs - Maths** which drills the student in addition, subtraction, multiplication and division. Under development is **Happy Programs - U.S. History**.

Best Electronics

0101 408 243 6950

Another supplier of general Atari 8 bit products that sell one product I have not seen anywhere else - replacement black

membrane pads for the Koala Tablet (\$15).

Black Moon Systems

P.O. Box 152, Wind Gap, PA 18091

Recently released **Labelmaster**, **Multi-Column Lister** (sold as package for \$10) and **Small Business System** (\$30). The first allows you to design labels with upto six lines of 34 characters. MCL allows you to print your Labelmaster files in one to four column format and SBS, which includes the other two products, is collection of programs designed to help a small business.

Computer Mail Order

0101 717 327 9575

CMO supply many different add ons from modems, printers and memory upgrades. Of particular interest to Atari 8 bit users are the interfaces; the **Supra 1150** (\$36), **Xetec Graphix Interface** (\$38) and **ICD P:R: Connection** (\$58). Memory upgrades available include the **Axlon 32K Memory Card** for the 400/800 (\$16).

Computer Software Services

0101 716 467 9326

The company that designed **Archiver** and **Super Archiver II** (\$99.95) has increased its product range to include the following products. The **Black Box** (\$199.95), which is being imported by Bruen Enterprises allows you to connect a hard disk, modem and parallel printer to you 8 bit. The **Multiplexer** (\$199.95) which allows you to connect 8 slave Ataris to one master that acts as a resource sharer for hard drives etc. BBS software is currently under development to use this configuration for a multi user BBS. The **SIO "Power Booster"** (\$39.95) amplifiers the signal from your computer by 20 to 100 times providing perfect square wave data transmission regardless of the peripherals used. The **Quintopus** (\$39.95) is a I/O port expander that converts a single I/O into five I/O ports, thus eliminating the need to daisy chain peripheral. A switchable version (\$59.95) is also available that allows two computers to be connected to the same printer, or two printer to the same computer. While one computer is printing you could be using another with a modem, disk drive or cassette. The latest craze at the moment is the use of removable hard drives, well CSS provide a **5Meg Removable Hard Drive** (\$495) for your 8 bit. The **Selector** (\$29.95) is a drive modification that allows you to change the drive number on a front panel switch, instead of using the slide switches located at the rear. The **Battery Backup!** (\$69.95) is a device (as its name applies) that maintains power during power blow outs. This device will probably not work within the UK, but does prove that it can be done. The **RAMdisk "Write-Protector!"** (\$29.95) is a hardware modification that will prevent accidental re-formatting or overwriting of RAMdisk. Works on all memory upgrades up to 2Meg. Other products include the **Ultra Menu / DOS** (\$29.95),

80 Column Richmonds' Word Processor (\$59.95), the Black Patch (\$49.95), the Klone II! (\$99.95), the Electronic Phantom Sector Maker (\$49.95), the Ultra Speed Plus (\$69.95), XF551 Enhancer (\$29.95), the Impossible (\$150), the Silencer (\$30), the Diskcracker (\$50), the Tranquilizer (\$29.95), the XL/XE Fix (\$49.95 ROM or \$39.95 disk or cassette) and Compare It (\$19.95). Most of these are hardware mods for disk drives allowing the backup of protected software.

Covox Inc.

Have released **Escape From Planet X** which is a text adventure that uses the **Covox Voice Master** or **Voice Master Junior**, which I assume are speech synthesiser.

DataQue Software

P.O. Box 134, Ontario, OH 44862

Turbo-816 (\$159.95 for kit), which is on import from Bruen Enterprises (01 368 4155), is a 16 bit CPU replacement for the CPU within the XL/XE range of computers. It is compatible with all XL/XE hardware and software and supports two modes of operation; 8 bit which uses the standard OS and 16 bit which uses Turbo OS. When in 16 bit mode an address space of 8 Meg is available without the use of bank switching.

Innovative Concepts

0101 313 293 0730

If you own a 8 bit with at least 128K of memory and fancy digitizing pictures then **Easy Scan II** (\$99.95) may suit you. Images can be scanned into either graphics mode 8, 9, 10, 11 or 15 and can be converted into any of the popular art package formats. System also requires an Epson compatible printer. A demo disk is available for \$5.

L & Y Electronics Inc.

0101 703 494 3444

A supplier that specialises in the ST but still offer a comprehensive range of serious 8 bit software and hardware including the Diamond GOS cartridge, Newsroom and the XEP80.

Logic One

P.O. Box 18123, Cleveland, OH 44118-0123

Two useful utilities for heavy RAMdisk users. **EDIT8** (\$9.95) allows you to sector edit the RAMdisk, while **The Enhancement** (\$7.95) lets you use your RAMdisk like a real disk. Working with Atari DOS 2.5 or SpartaDOS, the Enhancement lets DOS know when to recover and when to set up the RAMdisk, making it both automatic and coldstart safe every time.

Mars Merchandising

0101 312 627 7462

A company that mainly specialises in the Atari ST, but do sell the **Martian Atari Light Gun** (\$39.95) along with cartridges produced by Atari for their own light gun. MM also sell the Diamond GOS cartridge (\$69.95).

Micromiser Software

0101 904 383 0745

A company that specialise in the serious software market that have released three products for the Atari 8 bit. **Turbobase** is

flexible and powerful database complete with 600 page manual. **Turboword Plus** and **Turbofile** both produce a 80 column display using the XEP80 and are a wordprocessor and data/file manager respectively.

Newell Industries

0101 214 442 6612

NI offer a wide range of hardware add-ons for the complete range of 8 bit computers. For the old 400/800 they can provide the **Fastchip** (\$19.95) Floating Point ROM and **Omnimon** (\$29.95) piggy back board with **Omniview 80**. Also, just for the 800, there is the **RAMROD OS Board** (\$59.95) with **OSN OS** and **OMNIMON**. For the XL/XE variants there is **RAMRODXL** (\$49.00) which contains **Omnimon**, **Fastchip** and **OSNXL**. Their final OS upgrade is **Omniview XL/XE** (\$29.95) which is a 80 column operating system. They also provide the **SBM** (\$49.95) for the Atari 800 and **SBM130** (\$49.95) inventory, point of sale systems.

No Frills Software

800 East 23rd Street, Kearney, NE 68847

Users of Print Shop may be interested in the **PS Users Utility Disk** (\$31.95) which contains a set of utilities that will help maintain a catalogue of fonts, icons etc. Additional features allow you to mix Printshop graphics with Atari fonts to print custom labels and bookmarks.

San Jose Computer

0101 408 224 8575

Another supplier that specializes in the supply of hardware. Fancy a **1200XL**? Well, SJC are the only company I could find still selling them. Also available, disk drives, software on disk, cartridges and spares.

Wizztronics

0101 516 473 2507

Fancy controlling a MIDI keyboard from your Atari 8 bit, well Wizztronics sell the **MIDI Max MIDI interface** (\$179) which plugs into the Serial I/O. Hardware complete with port for daisy chaining. Comes complete with **MIDI Music System** software from Synthetic Software. Also available are 11 disks of ready to run songs (full set - \$55).

The following products were listed in a 'New and Improved' article within Atari Explorer without given the producers telephone number or address. **Desktop Performance Studio** by **Virtusonics** and **Guitar Wizard** by **Baudville**. Also two new bulletin board programs have recently been released; **Oasis BBS 4.2** from **Z Innovators** and **Carina II** from **Carina Software Systems**. If and when I obtain more details about these products I will let you know.

We are currently looking at the possibility of importing some of these products. If there are any items within this article, that are not currently being imported, which you would like drop us a line (BaPAUG; normal address) and we will try and get it (them) for you. Please mark your envelope US Products.

The Xtra-Ram Upgrade Board.

A review by Graham Jones

Many owners of Atari STs will, after an initial 'honeymoon' period with their new computers, start thinking of ways to improve their system. Unless you have already purchased a Mega ST2 or more, then a memory upgrade is likely to be fairly near the top of your list. However, you don't want to be without the computer any longer than necessary and a DIY upgrade seems an ideal move. A Yorkshire company by the name of Frontier Software advertise heavily in the computer magazines and their board has the added advantage that you can upgrade first from 520K to 1 Meg and later to 2.5 Meg when funds allow. The purpose of this review is to explain the whole process from ordering through to actual use.

Having made the decision to purchase the system, contact must be made with the firm. This is not quite so easy as one might have hoped. Although there are two phone numbers in the advertisements, they appear to be engaged with regular monotony! This must be a popular firm! However they do have an answerphone on which you can leave details for them to phone back the next day. I finally made contact and established that there were no retail outlets in the Bournemouth area, so elected to purchase by mail order, paying by our 'flexible friend'.

Dispatch was promised for next day, with recorded delivery. Perhaps I was unfortunate in that I ordered just after the 16-bit show, but in fact dispatch was held up for a further five days, although there was an apology on the dispatch note.

What then do you get for £100? The package, which was securely wrapped, consisted of a box containing the main XTRA-RAM board in an anti-static board. Two adaptors which have to be connected to the computer, both complete with ribbon cables, some small black jumpers, needed only for the 2.5 Meg upgrade, a disc containing the test program for the memory upgrade and a 33 page manual. All of this was well protected with polystyrene chips to prevent damage.

At this stage a thorough study of the manual is highly recommended. The various types of ST are covered from the early 520ST, STM and STFM. There is a warning that opening your computer will invalidate your guarantee and at this stage it is still impossible to tell whether you can actually do the upgrade yourself. The two crucial chips are the Video shifter and

MMU. These must both be in sockets, if not you cannot proceed. All descriptions here will refer to a STFM.

The manual claims that the board can be fitted in one hour. I had never opened my Atari before and I would allow between two to three hours to be on the safe side. Time to begin!

The first stage is to remove 10 screws from the base of the computer, including those holding the disc-drive in place. There are three different sizes and types which should be kept separate, ready for re-fitting. Once these have been removed and the computer turned back up the normal way, the top of the case may be removed. The keyboard is now exposed, ready for removal. A single plug connects it to the circuit board, once free, it may be removed. Take the opportunity to give the keyboard a good clean, mine was very dusty. The circuit board and disk drive are both encased in shielding which has to be removed. This shielding is generally in two halves, with metal twists holding it together. A pair of pliers will help to straighten these. The drive shielding is removed first, which then allows access to the back of the disk drive. Two cables are connected to the back of the drive and should be removed. This allows the drive to be removed and put safely on one side.

The main shielding can only be properly separated when it is completely out of the computer. To achieve this it is necessary to remove two screws which hold the power board to the base of the computer. The complete circuit board, with shielding can now be removed. Place the base away safely and carefully undo any remaining twists. At this stage the two crucial chips are exposed to view.

If your chips are soldered and not in sockets then you will be unable to proceed. Re-assemble the computer, contact Frontier software and they will collect the computer, fit the upgrade and return it, with a two day turn around for an all inclusive price of £40.

There are four different layouts of ST circuit boards, all explained in the manual. I found that the version I had was also moulded on the inside of the computer case. The Video shifter chip sits in its own metal box which has to be opened. The chip can then be CAREFULLY prised out from the socket. The adaptor board fits into the socket, with the chip fitting into a new socket on top. The cable is fed through the box and the lid closed down. The manual emphasises the care needed to avoid problems caused by static.

The MMU chip is a square chip, held in place on my board with a retaining clip, which must first be removed. This time the adaptor board has to fit on top of the chip and I found great difficulty in lining-up the adaptor with the socket. A relatively large force is needed to secure the board in place, tight against the chip. The cable from the adaptor board will have to be fed out through the shielding to reach the main RAM board. Some thought about which route will be taken may save time being wasted later. I re-assembled the shielding and then found that the cables would not reach the appropriate places on the RAM-board.

If all is well, then the shielding is re-assembled and the board screwed back into place in the lower half of the case. The disk drive is re-connected, now the RAM-board can be fitted. Two plugs connect the adaptor cables to the RAM-board, it is important to fit them in the right places and the right way round. There is also an earth lead, which has to be fixed to the power board shielding. The keyboard may now be replaced, the board may need to be manouvered to allow a good fit. Refit the top case, but before replacing the screws, I would recommend that you check the computer is working.

BaPAUG News

Meetings

The meeting schedule for the rest of the year follows, with the meetings being held on the 1st Friday of each month at the Kinson Community Centre, Bournemouth.

March	Hardware
April	Desktop Publishing and Word Processing
May	Public Domain
June	MIDI
July	SIGs Night
August	Graphics
September	Annual General Meeting.
October	Sound Sampling
November	SIGs Night.
December	Traditional Games and Chess Challenge.

It has been suggested that we hold a Xmas party in December, in addition to our monthly meeting. If anyone has any suggestions for a venue, theme etc, then please inform the committee, remembering that this will be an event for us and our families.

Rule Change

Rule 11: The possession of copied copyrighted software shall be prohibited during group meetings.

With the introduction of the Language SIG it soon became apparent that Rule 11 of the BaPAUG rules would prevent the use of some of the ST languages within group meetings. The same also applied to any application (Timeworks Desktop Publisher for instance) that required working disks to operate and couldn't be used from the disks supplied within the package. For those of you who missed the January and February meetings Rule 11 has now been changed to read as follows:

Rule 11a: Members shall only bring to meetings software of which they own the original.

Rule 11b: Working copies may be brought to meetings providing the original copy or original manual is also brought.

All members will be asked to sign a new declaration stating that they have read the rules, including the above change, and will abide by them.

The disc provided contains a RAM check program, which will check how much memory is now available. In my case all was well first time. There is also a memory testing program called TEST RAM, which should be left running for at least half an hour to check the RAM chips. The documentation of these programs is well covered in the manual. There are also several other programs supplied on the disk, including a Ramdisk, printer spooler and a program which indicates the memory map. The documentation of some of these programs appears to be non-existent. Room for improvement here!

In fact all the programs are quite useful once they are run, but it would be helpful to have some documentation on all the programs on a Read.Me file.

Frontier allow you to return the board for a full refund within ten days if you are dissatisfied, providing everything is in saleable condition. Take care how you open the boxes! Thereafter a 12 month guarantee is offered, providing no modifications have been made to the circuit board.

One further problem can occur with older STs, made in 1985 and early 1986. In some cases the MMU chip cannot access more than 512K of memory. The solution is to replace the MMU chip at a cost of £33.

Overall then, what's the verdict? Providing that you are able to fit the upgrade yourself, without any extra expense, then the XRAM board is good value for money at £99.99 plus £2.30 postage. There is the possibility of a later upgrade to 2.5 Meg, simply by removing the memory chips and replacing them with 1 Megabit DRAMs. If however you have to have the board fitted, or less likely, your MMU chip is too old then there may be more attractive alternatives.

If you have never opened your computer before, I would suggest that you set aside a whole evening to fit the board and run the test programs. Anything less is a bonus!

Next Issue

Within the next thrilling issue of 8:16 the following articles should appear:

Stereo Sound on your Atari 8 bit.

Introduction To C Programming.

Atari 8 bit : The Hardware and how it works.

Assembler Programming Techniques
for the Atari ST.

Multi Coloured Characters on the Atari 8 bit.

Using DOS 2.5 Part 3.

ST Educational Software.

Introduction to C Programming

By David Watson

In the first article I covered the layout of C programs, and described conditional statements and the looping constructs available to the C programmer. In this article I will build on this foundation. Arithmetical and logical operations will be covered in more detail, and functions introduced. This will serve as a lead into the topic of pointers. I will follow the same procedure as before; a task will be described, a pseudocode solution proposed and the C program displayed. Note that this will not be possible initially, so bear with me. Perhaps I should make it clear that I actually write and test the programs (using Sozobon C), and then import the source code directly into this article. This should mean that there are no errors in the programs.

Arithmetic Operators

To start this section, we will begin on safe ground and look at the operators which are common to other languages such as PASCAL and BASIC. They are addition, subtraction, multiplication and division. If you are using integer arithmetic then the modulo operator is also available. Consider the following arithmetic statements, together with a PASCAL equivalent. As I said before, the simple stuff first:-

C statement	PASCAL equivalent
<code>x = a + b;</code>	<code>x = a + b;</code>
<code>x = a - b;</code>	<code>x = a - b;</code>
<code>x = a * b;</code>	<code>x = a * b;</code>
<code>x = a / b;</code>	<code>x = a / b;</code>
<code>x = a % b;</code>	<code>x = a mod b;</code>

The last operator is the modulo operator. In C, integer division returns the answer of the division with the fraction truncated. The modulo operator effectively returns the 'remainder' of the division process.

Now let's look at some more interesting C arithmetic operators:-

C statement	PASCAL equivalent
<code>x = x + y;</code>	<code>x = x + y;</code>
<code>x += y;</code>	<code>x = x + y;</code>
<code>x += 1;</code>	<code>x = x + 1;</code>
<code>x++;</code>	<code>x = x + 1;</code>
<code>++x;</code>	<code>x = x + 1;</code>

Interesting, isn't it. The first statement is simple enough; add y to x. The second statement adopts a shorthand notation available in C, but performs the same task. The third statement is a repeat of the second, this time adding one to x. The final two statements introduce another shorthand notation, both adding one to x. However, there is a subtle difference between the two, illustrated in the following example.

Initially `y = 5, x = 6;`
`y = x++;` result is `y = 6, x = 7`
`y = ++x;` result is `y = 7, x = 7`

In each case the value of x is incremented by one. However, in the first case y takes the value of x before it is incremented, and in the second y takes the value of x after it is incremented. It is also possible to decrement variables using this technique; simply use a double-minus sign instead of a double-plus. It is not possible to apply any other operator in this fashion.

Fortunately though, there are several variants of the '+=' operator, and their use is illustrated below.

C statement	PASCAL equivalent
<code>x += y;</code>	<code>x = x + y;</code>
<code>x -= y;</code>	<code>x = x - y;</code>
<code>x *= y;</code>	<code>x = x * y;</code>
<code>x /= y;</code>	<code>x = x / y;</code>
<code>x %= y;</code>	<code>x = x mod y;</code>

This about sums up the arithmetic operators which are available to the C programmer, but before going on to discuss the logical operators, a word of warning. Consider the following two statements:-

`z = ++x; z *= y/x;`

It is not a good idea to write this as...

`z = ++x * y/x;`

Writing complex expressions like the above are something that we can all be good at, but when the time for debugging comes along we inevitably wish that we'd made things a bit simpler to understand. Another problem is that it is not safe to assume anything about the value of x for use

Operator	Associativity
<code>() [] -> .</code>	left to right
<code>! ~ ++ -- (type) * & sizeof</code>	right to left
<code>* / %</code>	left to right
<code>+ -</code>	left to right
<code><< >></code>	left to right
<code>< <= > >=</code>	left to right
<code>= = ! =</code>	left to right
<code>&</code>	left to right
<code>^</code>	left to right
<code> </code>	left to right
<code>&&</code>	left to right
<code> </code>	left to right
<code>?:</code>	right to left
<code>+= -= *= /= <<= >>= &= ^= =</code>	right to left
<code>,</code>	left to right

Table 1

in the division. You may find with your compiler that the value of x used is the result after incrementing, but other compilers will inevitably differ, and portability of code should be a concern with the number of C compilers which are available. It may seem disappointing to have to write the expression in two lines, but all is not lost; the little known comma operator comes to the rescue. Expressions which are separated by a comma are evaluated left to right and the result returned is that of the rightmost expression. Hence we could write the above as:-

```
z = (++x , x+y/x);
```

or

```
( z=++x , z+=y/x);
```

Either way you will find that the comma operator is something you very seldom see in people's code, which is a pity because it's really useful at times, particularly in 'for' loops.

Logical Operators

Well, enough of the arithmetic operators, what about logical operators? There are really two types of logical operator: those which are used in logical expressions (such as true AND true = true), and those which are used in conditional expressions (such as a==1 and b==1). The different types are referred to as logical and conditional operators respectively.

The logical operators are quite simple. There are only four; the four being AND (represented by &), OR (represented by |), EXCLUSIVE OR (represented by ^) and NOT (represented by ~). When a logical operator is applied, it takes its arguments as bit patterns and operates on them accordingly. For example, 5 & 3 = 101 (binary) & 11 (binary) = 1. The not operator returns the one's complement of the argument - i.e. all zeros become ones, and vice versa. Feel free to apply logical operators on any of the default types (except float or double).

There's more good news, though. As with the arithmetic operators there are shorthand notations, and the notations available are listed in the following list:-

<u>Expression</u>	<u>C expression</u>	<u>C shorthand</u>
x = x AND y	x = x & y	x &= y
x = x OR y	x = x y	x = y
x = x XOR y	x = x ^ y	x ^= y
x = NOT x	x = ~x	

Assembly level programmers will be pleased to find out that left and right shift operations are also available. To shift

PSEUDOCODE

```
Read number from user
Calculate square of number
Display square of number
END
```

C PROGRAM

```
int square();

main() /* Program to square a number using a function */
{
    int number, result;

    printf("Enter number to square : ");
    scanf("%d",&number);
    result = square( number );
    printf("%d is the square of %d\n", result, number);
    scanf("\n");
}

int square( x ) /* Function to square an integer */
int x;
{
    int y;

    y = x*x;
    return(y);
}
```

Example 1

a variable d0 by d1 places, the following notation should be used. Note that the C programmer may have any value for n as the compiler handles the rest, whilst the assembly programmer is restricted by the addressing mode used.

<u>68000</u>	<u>C expression</u>	<u>C shorthand</u>
lsl d0,d1	d0 = d0 << d1	d0 <<= d1
lsr d0,d1	d0 = d0 >> d1	d0 >>= d1

The conditional operators are similar to the logical operators; there is AND (represented by &&), OR (represented by ||) and NOT (represented by !). The difference is that the conditional operators treat their arguments in the following way; a non-zero variable is considered to be true, whilst zero variables are false. Thus you may quite safely write:

```
do{
    something;
}while( !finished ); /*ie. while NOT finished*/
```

In this case we could safely treat 'finished' as a flag which indicates when some task is complete. When it is zero then the task is not complete; when it is non-zero the task is over.

Finally, for this part, Table 1 is a table of precedences. The table is listed in order of decreasing precedence; the associativity indicates in which order they are evaluated.

Functions

That covers the basics of arithmetical and logical operators; now for something more interesting - functions. To define a new function we use the same 'notation' used to define main(). Going back to the last article, the first example was a program which




```

long square( x ) /* Declare the function as returning a 'long' */
int x;          /* The parameter passed is an integer */
{
    long a, b;   /* Two local variables are required, both longs */

    a = (long)x; /* Convert the integer to a long */
    y = x * x;   /* Calculate the result */
    return( y ); /* Return the result */
}

```

Example 2

found the square of a given integer. The program within Example 1 does the same; it prompts the user to enter a number, then prints the square.

Now let us examine this in more detail. Firstly, the function 'main()' has changed very little. All that has been done is that a line has been added to put the result of the operation into the variable 'result'. This is then displayed using the 'printf' command. Now let us look at what is required in order to include the 'square()' function.

Firstly, it is necessary to declare the function, and this is done at the start of the program with the line, "int square();". Basically this tells the compiler that there is a function called 'square', which will return an integer. It is not strictly necessary to do this, as if the compiler comes across a function that has not been declared then it assumes that the function returns an integer, but I declared it for the purposes of example. If I was being very strict about it then I would have declared 'main()' as a function returning an integer in the same fashion. It is not possible to return anything from main() other than an integer.

When we come to writing the function, we need to declare it as a function returning an integer again. Again we do not need to expressly state that it returns an integer, but I did so for the purpose of example.

The list of variables passed to the function are enclosed by parentheses after the function name. The variables in the list should be in the order that they will be passed to the function.

Following this list of variables is a list of definitions, which declares the passed variables' types. The actual body of the function then follows and it follows similar rules to the function 'main()'. Variables are declared first, then comes the function body. If it is necessary to return a value as it is in our function, then this is done using the 'return()' statement, where the value to be returned is enclosed by the parentheses. The only exception to this rule occurs if we are exiting from 'main()' and wish to return a value (i.e. an integer). We should use 'exit(x)' in this case, where x is the integer we wish to return.

So, in summary, consider this function which will calculate the square of the integer argument passed to it, and return a long (see Example 2).

A well commented piece of code. In fact this is over-commented, but never mind. However you can see that the function follows the rules. Firstly the function is defined as returning a long, and having one parameter. This parameter is then declared as an integer. The body of the function then follows. If you wish to use this

function in a program remember to declare it at the start of the program with the line "long square()". As an aside, there is no rule which states that 'main()' must appear first, but since the compiler expects this then it's a good idea to put it first.

There is a new operation performed in this program, and it is one which gives C quite a lot of power when used correctly. This is the process of converting a variable of one type into a variable of another; in this case we convert an integer into a long. This process is known as 'casting'. Where we have a variable of 'type1' and we wish to convert it into a variable of 'type2', we can force the compiler to do this by writing the variable name preceded by the type we want to use. In the above case we have an integer, and we wish to manipulate it as a long. Hence:-

```
a = (long)x;
```

If it had been necessary to cast 'x' as a float, then we would have put the word 'float' in the parenthesis. Using this process it is possible to convert any of the default types (i.e. not user defined ones), including pointers to any other type. And casting to and from pointers lets you do all sorts of useful things that you can't do in PASCAL. It also creates scope for writing convoluted code that nobody can understand.

In the next article I will cover the topic of pointers. Pointers are a difficult thing to get the hang of, so I intend to devote some time and space to the topic. This will also cover the topic of how to return more than one parameter from functions, a topic which I have avoided in this article in order to give it a more in depth examination in the next.

I am racing through the basics just now, so apologies if you think that I'm being too superficial. However, if you have any problems, comments on my articles or questions then feel free to write to me at the following address. I can only provide what you want if you give me feedback, so I look forwards to hearing from you.

David Watson,
22 Lydbury,
Bullbrook,
Bracknell,
Berks.
RG12 3HH

SPEED RUN

Reviewed by Ian Brooker

Pull on your driving gloves, put on your helmet and climb into your Sierra RS Cosworth. Can you qualify for the RAC Rally?

Well, that's the gist of it anyway. Lets start at the beginning and load the game in O.K. the game has loaded, you start by selecting manual or automatic gears using the option key, followed by start to move on. Stick to automatic for now, its easier. Disk users will now be given three choices - a 2.3 injection engine giving you more speed, extra grip tyres (recommended for beginners) and a recovery crew which will save you valuable time if you crash.

Cassette users now rejoin us as we behold a wonderful sight - near ST quality graphics (this is really what you're paid your money for, as will become clear). You are given a behind the driver view of the inside of the car with the driver at the wheel and ready to go. On your left there is a clock showing your course time, with the gear lever in the centre. At the top of the screen is a mirror which serves only cosmetic use and four marks representing the left, right and centre of the road, along with your relative position to them. These will become more and more useful as your driving experience improves. Along the bottom of the screen is displayed your road speed and current gear selection. Ahead of you, through the front

screen of your car is a flat uninteresting landscape with a Pole Position type road.

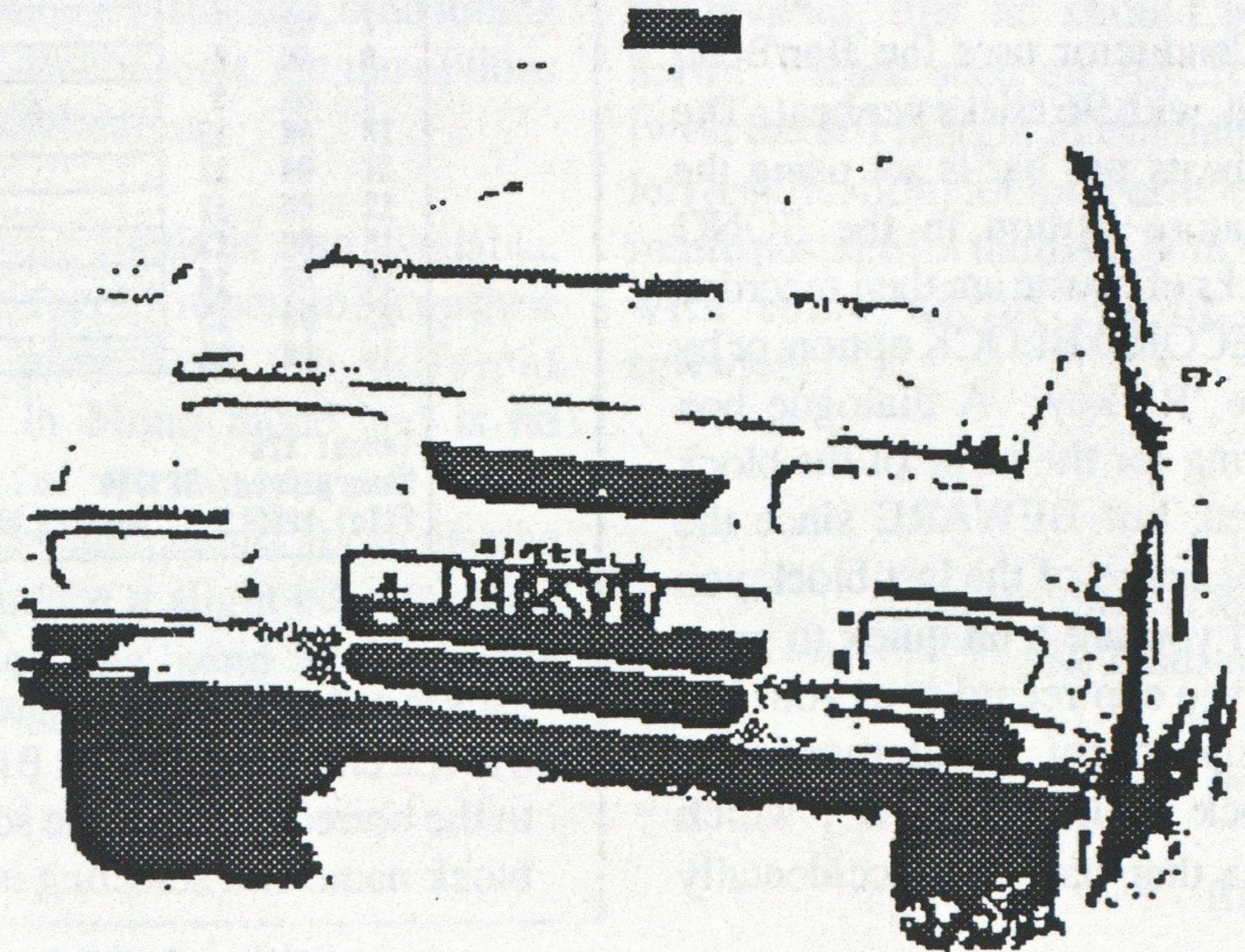
Push forward with the joystick and the engine fires into life, keep it forward to accelerate and the road moves jerkily past you. Push the joystick to the left and right to steer and watch the driver turn the steering wheel either way as you do so. The road gives you no details as to where you are, so keep watching those marks at the top of the screen. More driver movement comes into play when he changes gear. In fact, all animation of the driver is smooth and well timed.

Once you have mastered the art of driving around the course and you feel that your time cannot be improved, try it again using a manual gear changes, its a whole new ball

game. If you have the disk version try swapping between the 2.3i engine, the extra grip tyres and the recovery crew to see which helps you get round the fastest.

Overall, the graphics and animation are excellent inside the car, but not outside. The sound is disappointing and the game play limited to mastering the controls. Overall the game is reasonable value for money, especially if you want to show your Atari of to your mates, but don't let them play it for long.

Price: Cassette £8.95
Disk £12.95
Graphics: Inside car excellent, the rest average.
Sound: Poor.
Playability: Limited, especially the cassette version.
Value: Limited



Music Scene: MIDI

Super Conductor by MichTron.

A new series by
Paul Brookes

Super Conductor is a 16 track, budget sequencer aimed at the first time ST-MIDI user who wants an easy to use package, but isn't afraid of number-crunching! The latter point refers to the edit mode of Super Conductor, which can be a little tedious at times.

The program comes on one single-sided disk, and is accompanied by a 54 page, A5 size, leaflet-style manual, containing full and easy to follow instructions on how to use the program, a copy of the MIDI specification, and a 3 page index.

The Manual

I have found the manual easy to follow, but Super Conductor is so user friendly that after an initial reading, my copy has remained in its box. Appendix 1 in the manual describes how to program the internal sound chip of the ST. This is a facility I have never used, and if I ever wanted to program the ST's sound chip, then I would undoubtedly use Activision's "The Music Studio". Incidentally, if you own a MIDI instrument, or enjoy playing with the ST's sound chip then you must buy The Music Studio. If you shop around you can get it for as little as 10 pounds.

Super Conductor Basics

Ten songs of upto 16 tracks each, may be stored in memory at any one time, providing, of course, your ST has enough free ram. For a given song, each of the 16 tracks may be turned "on" or "off" and can be assigned to play and record on any available MIDI channel. A solo mode is offered allowing the user to hear one track on its own, without individually turning off all unwanted tracks. The right mouse button selects solo mode when the pointer is over the required track. The left mouse button restores the original set-up.

Super Conductor uses the Bar/Beat/Click format, with 96 clicks per beat. The number of beats per bar is set using the Time Signature option in the SONG menu. Blocks of music are then recorded using the RECORD BLOCK option, or by pressing the 'R' key. A dialogue box appears asking for the name of the block to be entered, but BEWARE since the default is the name of the last block you recorded. If you are a bit quick to press return then you can record over your last block! At this point I will mention a special block called 'UNDO', which restores data that has been accidentally

destroyed. Using the COPY BLOCK command, the UNDO block can be copied to any other block, such as one that has accidentally been over-written. Phew!

Block Operations

I have already mentioned the UNDO block. Blocks may be mixed, copied, appended, split, filtered (to remove certain MIDI data), transposed, quantized and edited.

Editing Blocks

In the edit mode, the note name, denoted by a letter and the MIDI number, the ON velocity, ON time (in B.B.C. format, see above), OFF velocity and OFF time, are displayed line by line. Program changes, polyphonic aftertouch, channel pressure, pitch bend and other controller information are also displayed together with the Beat/Bar/Click on which they were received.

Everything is easily edited, but to hear

edits you must leave the edit mode and return to the main screen (unlike in PRO-12, for example where notes may be played during editing). The method of editing is a little number intensive, but it can often be more flexible than the score edit mode of Steinberg's Pro-12, for example (see a future issue for a review of Pro-12/24, if I can figure out how to use them!).

Overall I must confess to liking the method of editing employed by Super Conductor. However, I have experienced unexpected crashes whilst in this mode, so saving data before editing is recommended.

System Exclusive Mode

If you have enough data on your synth's system exclusive mode then you will probably find this facility extremely useful. I haven't used it since I have no data on CASIO system exclusive commands, (if you have some please send

The screenshot shows the 'Main Song Screen' for 'Song 4: Yeh!'. The interface includes a menu bar (Desk, File, MIDI, Song, Block, Edit), a track list on the left, and a piano roll area on the right. The track list shows 16 tracks, with tracks 1-3 set to 'ON' and tracks 4-5 set to 'OFF'. The piano roll area displays notes for tracks 1 and 2, with track 1 containing drum notes and track 2 containing symbol notes. At the bottom, there is a status bar showing 'Tempo: 120', 'Memory Free: 589710', and 'File: \PAULS_SO.N65\YEH.MU'. A control bar at the bottom right contains buttons for 'Record', 'Play', 'Sizing', 'Load', and 'Save'.

Track	Channel	Notes
1	ON 1	drum drum drum drum drum drum drum drum drum
2	ON 2	symbol symbol
3	ON 3	
4	OFF 4	
5	OFF 5	
6	ON 6	
7	ON 7	
8	ON 8	
9	ON 9	
10	ON 10	
11	ON 11	
12	ON 12	
13	ON 13	
14	ON 14	
15	ON 15	
16	ON 16	

The Main Song Screen

This is one of the ten song screens. Each track is either on or off, and is assigned a MIDI channel number. Blocks may be inserted by dragging the required block to the correct point in the song, or by clicking the left mouse button, typing the block name and selecting insert.

Conclusions

I have used Super Conductor for two years now, and have found it extremely competent in nearly all respects. The B.B.C. format takes a bit of getting used to, but once mastered, allows very precise editing. Recently, I purchased Pro-12 which I am still getting used to. I prefer the graphical song representation of Super Conductor's blocks, to the method used in Pro-12. (Pro-24 has a grid edit mode which was left out of Pro-12 in favour of the score edit mode - more in a later issue). In short Super Conductor is a good sequencer for the computer user come MIDI dabbler, but real musicians will probably prefer a sequencer such as Pro-12 (or Pro-24).

In this mini review I have introduced some of the features of Super Conductor which I feel characterise the program, whilst leaving out some of the more basic commands such as Time Signature, Tempo etc. For those readers interested in seeing Super Conductor or either of the Steinberg sequencers in action, why not come along to the next MIDI meeting at the club. No dates have been fixed yet but the more interest shown, the sooner one can be arranged.

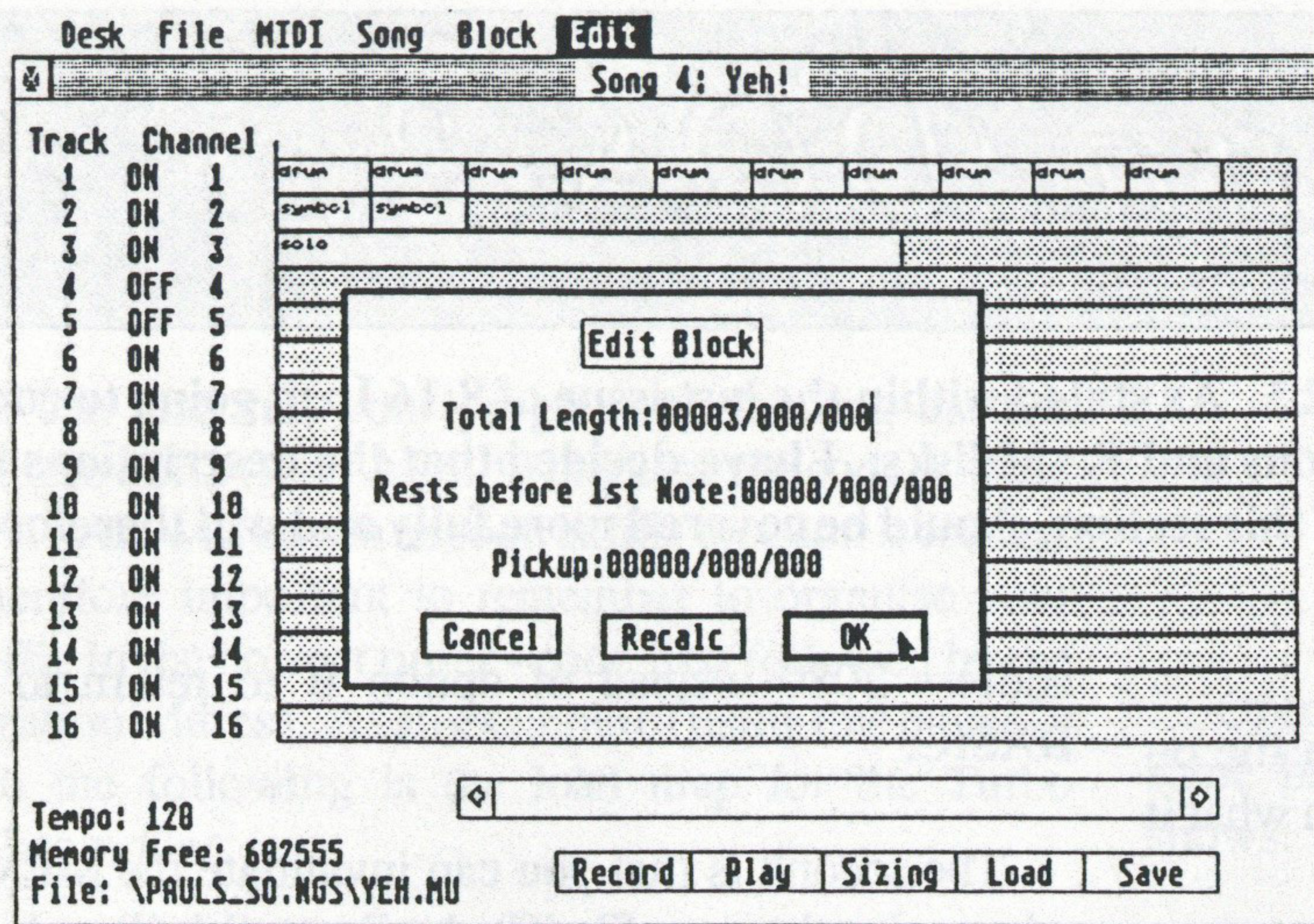
Comments

If you use Super Conductor and feel I have left out any crucial features of this budget program, or you have any further praises / groans about the software, then please write in!

Articles

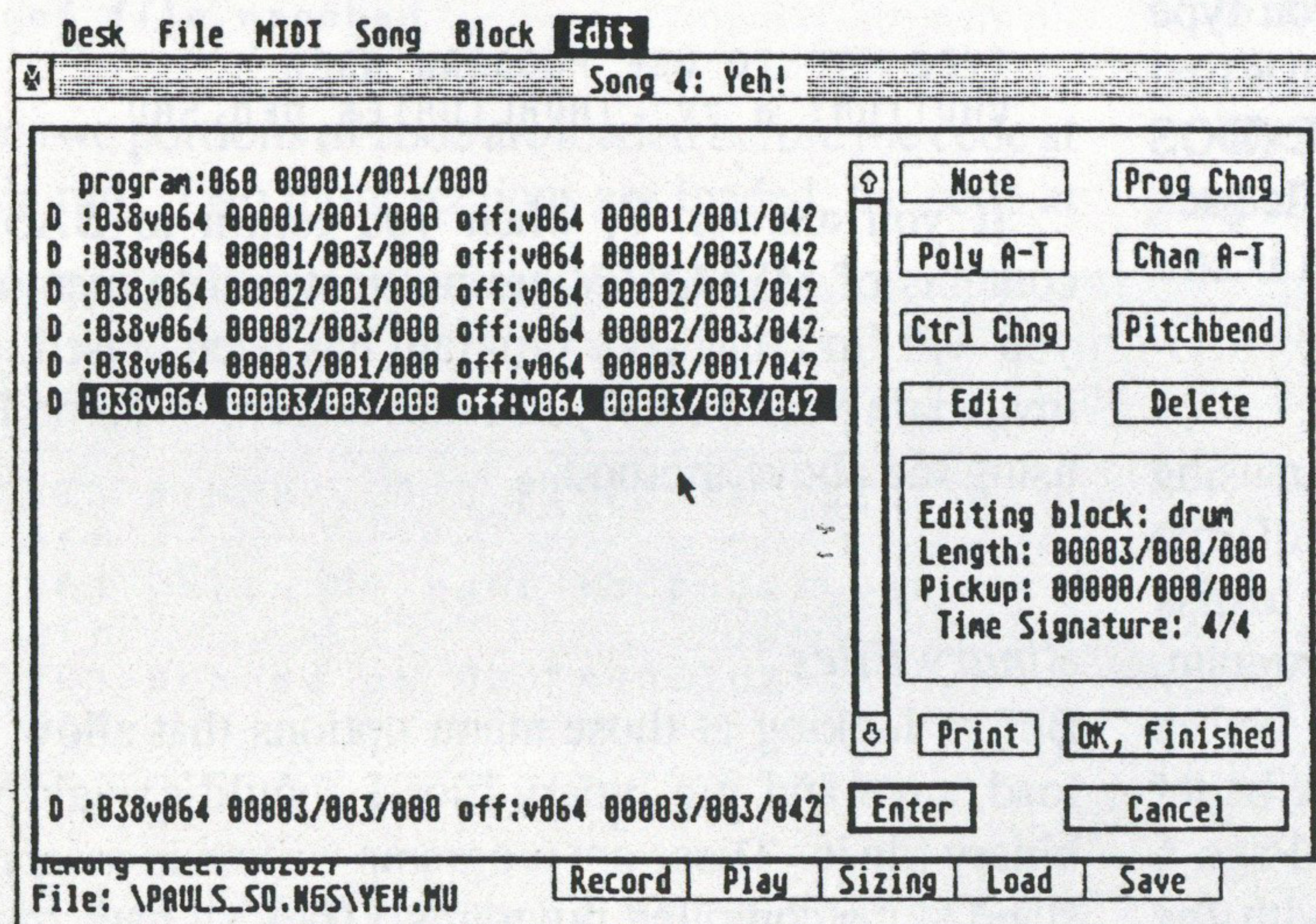
If you own a sequencer and feel you could write a review or some comments / guidelines on its use, write to 8:16 or directly to me at the address below. (Note from editor: Paul only owns a ST so any 8 bit reviews, tips etc should be sent to 8:16). Please send an ASCII disk file (with the left margin at column zero and left justification) for long articles, and the return postage for the disk, which I will fill with some useful Public Domain software.

Paul Brookes (Dept. 8:16),
32 Dudsbury Road,
West Parley,
Wimborne,
Dorset
BH22 8RE.



The Edit Dialog Box

This is the dialog box which appears when the Edit Block function is selected from the main menu. This allows the length (in Beat Bar Click format), pickup point, and rests before the first note recorded in the block to be set before detailed editing. This is often all that is required to remove any annoying pauses at the start and finish of a block, thus allowing the block to be looped or concatenated.



The Main Edit Screen

Here the notes and MIDI events are listed down the left hand side of the screen. At the bottom is an edit line which holds the event currently selected for alterations. This line is also used for the step time input of data.

it to the club as it will be gratefully received and might encourage me to produce some PD software for the CZ's).

To use the system exclusive mode, you must create a file with the '.EXC' extension, using any text editor such as TEMPUS (2), or 1st Word in ASCII mode. Chapter 6 of the manual is devoted to the use of the system exclusive buffers, and how to generate '.EXC' files.

MIDI Commands

Timer sync, active sensing, song number, song position, all notes off, all notes off simulation (for keyboards not responding

to the MIDI all notes off code), release pedal (controller 64- sustain), send tuning request and system reset, are the options under the MIDI menu.

An echo back mode is also available. This is useful for CZ owners to set up their synths to play 8 or 16 different instruments in Mono mode and is no doubt useful to users of other manufacturers equipment. It can also be used to introduce a slight echo to notes played when the send and receive channels on the synth are set the same.

Using DOS 2.5

Welcome to the second part of Using DOS 2.5. As stated within the last issue of 8:16 I am going to cover the use of MEM.SAV, those advanced DOS functions and RAMdisks. I have decided that the descriptions for the error messages, originally intended as a part of this section, should be covered more fully and will therefore form part 5 of this series.

What is MEM.SAV?

To answer this question bluntly, MEM.SAV is a file on your disk. This isn't very useful so let me explain what it does for you.

Whenever you need to go to the disk utility package, say by typing DOS from BASIC, the file DUP.SYS is loaded into low memory. Your BASIC program is also stored starting in low memory and is therefore over written by DUP.SYS and destroyed. To prevent this from happening you create a file on your boot disk called MEM.SAV using DOS option N. Now when you type DOS the portion of memory that will be overwritten is first saved to disk. When you wish to return to BASIC (DOS option B) this saved data is then copied from this file back to memory. Thus when you return your program is still intact.

There are two things you should remember when using MEM.SAV. The first is that this doesn't work with Turbo BASIC, thus meaning you cannot jump from basic to disk utility package and back and still preserve your program. This can be overcome by saving your program before typing DOS, though if you forget its too late. A better solution was published in the update section of Page 6 (issue 38). Using MEM.SAV, use option M with the

address 2080 instead of option B to return to Turbo BASIC.

The second is that you can invalidate the MEM.SAV when using the copy file (C), duplicate disk (J) or duplicate file (O) options. When you have an active MEM.SAV file DOS knows that it cannot use the memory where the rest of your program is stored, thus limiting the amount of storage space and possibly increasing the number of passes. Thus when you use any of these option you are asked:

```
TYPE "Y" TO USE PROGRAM AREA
CAUTION: A "Y" INVALIDATES MEM.SAV
```

If you answer Y, when you return to BASIC the contents of MEM.SAV are not restored to memory and you will find that your program has been erased. If you invalidate MEM.SAV you cannot return to Turbo BASIC using the above method.

Binary Files

Before looking at those menu options that allow you to load, save and run binary files I should explain what a binary file is. These are programs written in assembler or stand alone compiled programs. They all have one thing in common in that you can run them without any additional software being present, such as your BASIC cartridge. To make sure that the file being loaded is a binary file the operating system checks the file header which is six bytes long. If the first two bytes are \$FF and \$FF, the file is a binary file. Any other values will result in the error message BAD LOAD FILE (from the DOS menu) or a load error and possible crash if loading at boot. The next two bytes tell the operating system where in memory the file should be loaded, while the 5th and 6th byte tell the OS the number of bytes to load. When the OS has completed this transfer it then attempts to read another header to see if there is any more to load. This repeats until the End Of File has been reached. It should be noted that any subsequent header do not need the \$FF / \$FF identifier, though they may be present.

Menu Option L: Load Binary File

This menu option allows you to load (and run) a binary file. The load sequence follows exactly the one

The Menu Display

The following diagram shows the menu display when first entering the disk utility package.

```
DISK OPERATING SYSTEM II VERSION 2.5
COPYRIGHT 1984 ATARI CORP.
```

```
A. DISK DIRECTORY      I. FORMAT DISK
B. RUN CARTRIDGE      J. DUPLICATE DISK
C. COPY FILE          K. BINARY SAVE
D. DELETE FILE(S)    L. BINARY LOAD
E. RENAME FILE        M. RUN AT ADDRESS
F. LOCK FILE          N. CREATE MEM.SAV
G. UNLOCK FILE        O. DUPLICATE FILE
H. WRITE DOS FILES   P. FORMAT SINGLE
```

```
SELECT ITEM OR RETURN FOR MENU
```


describe above. Upon completion of the load the OS jumps to the memory location pointed to by \$2E0 (RUNAD). This will normally pass you back to the DOS menu, but can be used to run your program if you load in your own run address.

You can also run your program by loading the run address into \$2E2 (INITAD). In this case the code pointed to by INITAD will be run as soon as this location is loaded. It is therefore important to remember to organise your binary file in the correct order - code first followed by the initialization address. As an example of using this memory location the following is the load map for the Turbo BASIC compiler:

```
Load address = $3E00; Length = $80
Load address = $3000; Length = $0163
Immediate run address = $3000
Load address = $207D; Length = $011C
Load address = $6000; Length = $010E
Immediate run address = $6000
Load address = $C000; Length = $0B3E
Load address = $E400; Length = $1528
Load address = $DE00; Length = $0200
End of file reached
```

Here two portions of code are loaded before the code at \$3000 is run. Two more sections are loaded, the code at

\$6000 is run and the loaded completes with the loading of the final three sections. By building your binary files in this way it is possible to display title screens while the bulk of the program loads, just as Turbo BASIC does when it displays:

```
*Happy_*
Computer
```

```
TURBO-BASIC XL 1.5
(c) 1985 Frank Ostrowski
```

Of course you can use RUNAD and INITAD within the same binary file, as the load map for COPY32.COM shows:

```
Load address = $0300 Length = $12
Immediate run address = $0300
Load address = $3400 Length = $1AB5
Run address = $4E7A
End of file reached
```

(Note: COPY32.COM is the utility supplied with DOS 2.5 which allows you to convert your DOS 3.0 files into DOS 2.5 format)

The load maps shown were produced using the program shown in Listing 1. Version 2 of this

Listing One

```
100 -----
110 REM BINARY LOAD FILE ANALYSER
120 REM
130 REM Colin W. Hunt ; DaPAUG 1988
140 REM
150 REM Stored as OBJEXAM.TUR
160 REM
170 -----
180 GRAPHICS XO:POKE 752,X1
190 CLR :CLOSE
200 ? "      Binary Load File Analyser"
?
210 POKE 85,11:?" by Colin W. Hunt":?
220 DIM A$(X1),ADDR$(4),FILENAME$(15),
DESC$(35)
230 Z=DPEEK(741)-DPEEK(144)-550:DIM F$(
2)
240 * GET_NAME
250 ? :? "Enter OBJECT filename":INPUT
"<D:filename.ext> ";FILENAME$
260 IF LEN(FILENAME$)<X3 THEN GO* GET_
NAME
270 IF FILENAME$(1,2)<>"D:" THEN GO* G
ET_NAME
280 TRAP *NO_FILE:OPEN *X1,4,X0,FILENA
ME$:TRAP 40000
290 GET *X1,A:GET *X1,B
300 IF A<>255 OR B<>255 THEN CLOSE :?
"NOT BINARY FILE":? "Press any key":GE
T K:GO* GET_NAME
310 * GET_OUTPUT
320 ? :INPUT "Output to Printer or Scr
een ",A$
330 IF A$="P" OR A$="p"
340 TRAP *NO_PRINTER
350 OPEN *X2,8,X0,"P:"
360 TRAP 40000:OUTPUT=X1
370 ? :? "Enter description (<35 Cha
rs)"
380 TRAP 380:INPUT DESC$:TRAP 40000
390 ELSE
400 OPEN *X2,8,X0,"S:"
410 POKE 752,X1
420 ENDIF
430 ? *2:DESC$
440 FOR I=X1 TO LEN(DESC$):? *X2;"-":;
NEXT I:?" *2
450 TRAP *END_OF_FILE
460 DO
470 EXEC GET_ADDR:PC=ADDR
480 IF PC=$FFFF THEN 470
490 EXEC GET_ADDR:FIN=ADDR
500 IF PC=736
510 EXEC GET_ADDR
520 ? *X2;"Run address = $";HEX$(A
DDR)
530 ELSE
540 IF PC=738
550 EXEC GET_ADDR
560 ? *X2;"Immediate run address
= $";HEX$(ADDR)
570 ELSE
580 IF PC=743
590 EXEC GET_ADDR
600 ? *X2;"Low memory pointer
= $";HEX$(ADDR)
610 ELSE
620 ? *X2;"Load address = $";H
EX$(PC);" Length = $";HEX$(FIN-PC+X1)
630 BGET *X1,ADR(F$),FIN-PC+X1
640 ENDIF
650 ENDIF
660 ENDIF
670 LOOP
680 STOP
690 -----
700 PROC GET_ADDR
710 GET *X1,C:GET *X1,D:ADDR=D*256+C
720 ENDPROC
```


Listing One Continued

```
730 -----
740 * END_OF_FILE
750 IF ERR=136
760 ? *X2;"End of file reached"
770 IF OUTPUT=X1 THEN ? "End of file
reached"
780 ELSE
790 ? "ERROR ";ERR;" on line ";ERL
800 ENDIF
810 CLOSE :POKE 752,XD
820 END
830 -----
840 * NO_FILE
850 TRAP 40000
860 CLOSE
870 IF ERR=130
880 ? "Unknown device."
890 ELSE
900 IF ERR=138
```

```
910 ? "Device does not respond."
920 ELSE
930 IF ERR=170
940 ? "File not found"
950 ELSE
960 ? "ERROR ";ERR;" on line ";E
RL
970 ENDIF
980 ENDIF
990 ENDIF
1000 GO* GET_NAME
1010 -----
1020 * NO_PRINTER
1030 TRAP 40000
1040 CLOSE *X2
1050 IF ERR=138
1060 ? "Printer does not respond."
1070 ELSE
1080 ? "ERROR ";ERR;" on line ";ERL
1090 ENDIF
1100 GO* GET_OUTPUT
1110 -----
```

program can be found on our PD Disk #14 : The Turbo Collection, which also contains other useful utilities and demos written in Turbo BASIC.

Menu Option M: Run At Address

Back to DOS 2.5. As stated, menu option L allows you to load one of these binary files, with there being three possible outcomes. One, the program runs. Two, the program crashes or the system crashes (usually caused by the binary file overwriting DOS) and three, the OS returns to the DOS menu. It is this latter option we will now consider. So how do you run the program if it didn't run itself? Answer: use DOS menu option M. The only snag is you need to know the run address, though you can usually assume that the first location loaded is the run address. When you enter the run address at the prompt

```
RUN FROM WHAT ADDRESS?
```

remember that DOS is expecting a HEX number.

Menu Option K: Save Binary File

This option allows you to save an area of memory in a binary file form. Upon selection the following prompt will appear:

```
SAVE-GIVE FILE, START, END(, INIT, RUN)
```

The minimum requirement for this option is FILE, START, END. The parameters START, END, INIT and RUN should all be entered as hexadecimal numbers, whereas file should be entered in the form Dn:fname.ext. If you do not include the INITialization and RUN addresses when you load the file using option L it will not be run. Instead you are returned back to the DOS menu. If you only specify the INIT address your program will run after loading and you will still be able to return to the DOS menu by pressing RESET, as this will still be vectored via the RUN address. If you wish you program to re-initise

after RESET set RUN the same as INIT.

RAMdisks

If you have a 130XE or a XL/XE that has a memory upgrade taking it upto a 128K of RAM you can, if you wish, create a RAMdisk within the second 64K bank when using DOS 2.5. When you boot your system it determine if this extra memory is available, if yes, it then loads the RAMDISK.COM which contains the necessary software to initialise and maintain it. When initialised the RAMdisk, which will be drive 8 within your system, will contain 499 free sectors, of which, some will be used to contain the files DUP.SYS and MEM.SAV thus making the movement from BASIC to the disk utilities quicker. To do this the software modifies DOS in order that it will look for the DUP.SYS file \on drive 8. If you delete DUP.SYS from your RAMdisk you will also have to change location 5439 to point to the appropriate drive; eg POKE 5439,ASC("1").

The load map for the RAMDISK.COM file, for interest, is as follows:

```
Load address = $3800 Length = $041E
Run address = $3B5D
```

Conclusion

That wraps up this instalment on DOS 2.5. If you have any questions regarding anything within this article, or the previous, feel free to drop me a line. You will find my address on page 3. Within the next issue of 8:16 I will explain the uses of those extra utility programs found on the DOS 2.5 disk : SETUP.COM, COPY32.COM and DISKFIX.COM.

Subscribe to 8:16 and get the complete DOS 2.5 and Turbo BASIC software packages on our free subscription disk.



NEW!

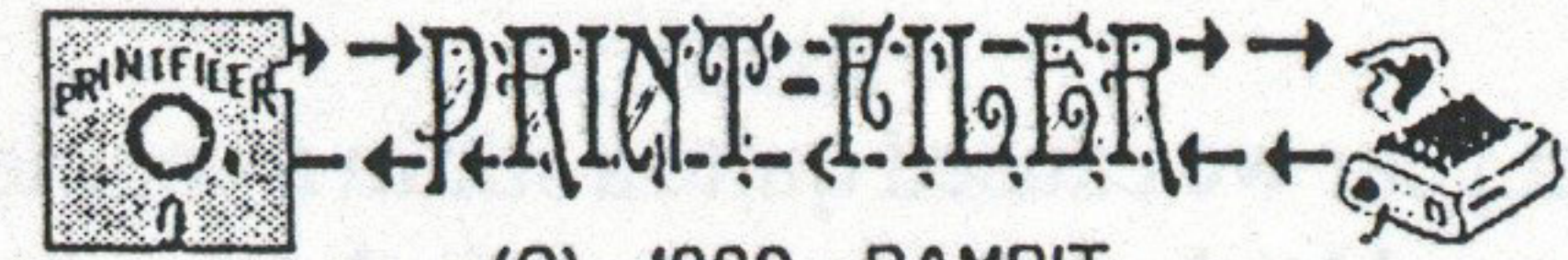
PRINTER USERS Enhance your output, select & combine Graphics from different sources to print in text or alone as desired ie. letter heading with your letter. PRINT-FILER can directly produce disk files from such as BRODERBUND PRINTSHOP, which will then print without the parent software, also includes EDITOR for customising of files. Disk £8.

FAST TAPE LOADING Cut times by 5/6ths. approx, fit RAMBIT to your XC12 and use RAMBIT II to convert machine code games up to 48K. and TURBO-BASIC (public domain, disk only till now). TURBO-BASIC Loads in 75 sec.s compared to 35 from disk, is totally compatible with normal Atari Basic adds powerful commands and runs programs written in it or Atari Basic at 3 to 4 times normal speed. KIT + RAMBIT II & T.B.(state cass. model) £16. Fitting +£7 (post cass. to us). Also ASSEMBLER/ EDITOR for RAMBIT II. Tape £6.

TAPE TO DISK Put your machine code tapes on Boot disk with Menu selection; successful utility for learning and experienced drive users, DOS 2.5 compatible, has assembler editor facilities. TASKMASTER Disk £9.

All products require minimum 64K XL or XE and come with full instructions. Prices are all inclusive. For details send S.A.E.,or P/O, Cheque payable to:- RAMBIT 16 THE GREEN,THURLBY, BOURNE, LINCS. PE10 0HB.

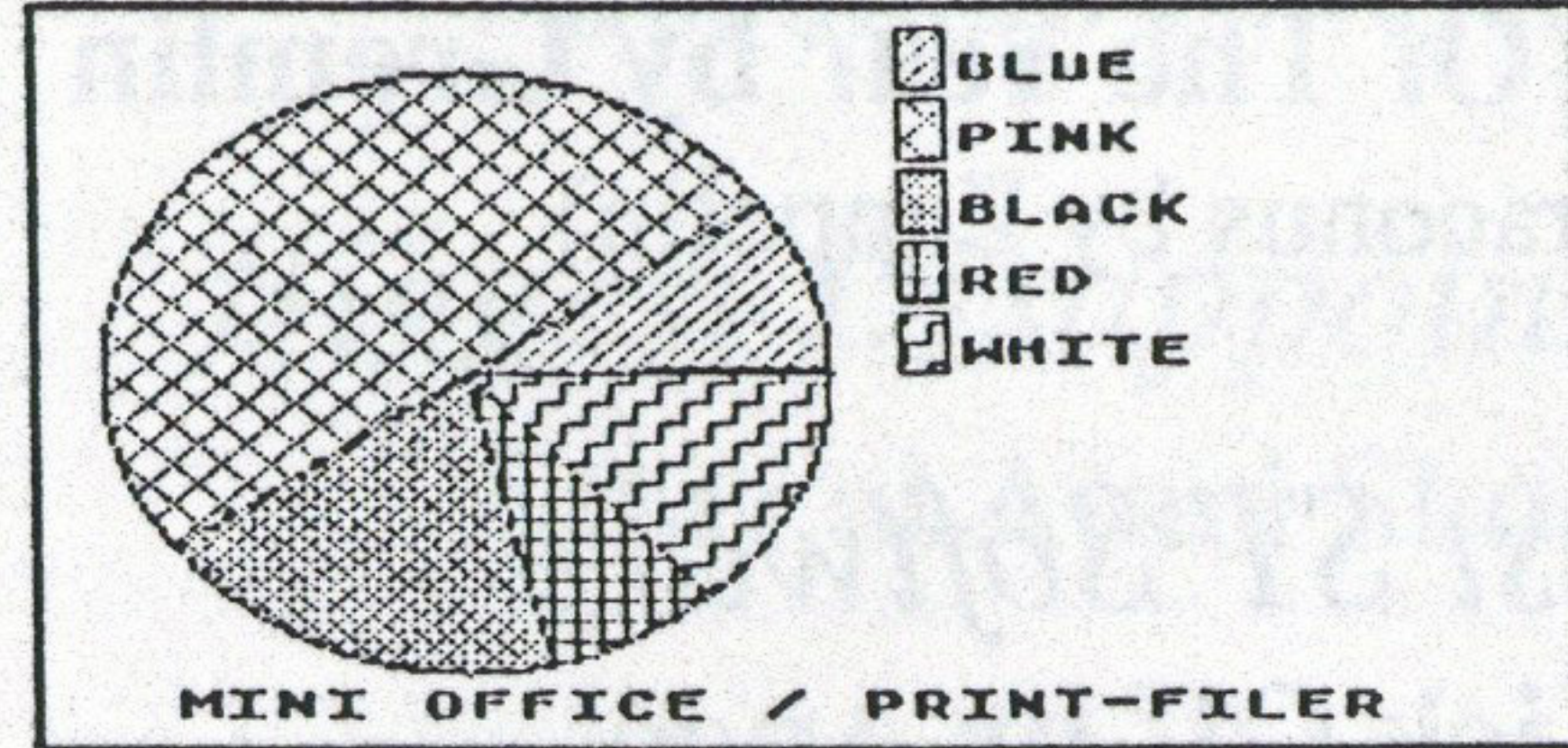
EXAMPLE PRINT-FILER PRODUCTION



(C) 1989 RAMBIT

When working with print utilities such as the excellent PRINTSHOP by Broderbund do you find yourself wishing to combine output of more than one routine or even utility to produce a particular effect, or inconvenienced at having to print fancy work and word process output with separate passes? We at Rabbit did and so designed PRINT-FILER to fill this void in 1 bit printer utilisation.

PRINT-FILER is loaded before your selected utility, BASIC or otherwise, and takes over at the printing stage of your work to allow you to either print as normal or save to disk in DOS2.5 format; how does this achieve anything extra you may ask. Simply this, the disk file produced contains ALL the information required to print your workpiece without reference to special font or utility- so by transferring the file to printer ie. with "C" in DOS2.5 or "TPD:(filename)" in MINI OFFICE !! wordprocessor or even via a Modem to a different computer you can print it, ie. this sheet comprises 5 files saved as one file with a second pass of P-P. so that copies can be printed at maximum speed from memory using the editor.



Also provided with PRINT-FILER is an EDITOR specifically designed to enable you customise your files such as combining sections of different files, repositioning of icons etc. Documentation is included on the disk to be printed using the editor or DOS2.5 "C" option. PRINT-FILER loads to the switched RAM under the operating system ROM, has been tested and found to work well with many programs such as used to produce spreadsheet, graph, label etc, requires a standard XL or XE with 1 or 2 disk drives (110 or 1050 3D or DD) and any printer which will work via the Atari serial connector. The only limitation in this respect is in editing of the output as some options are specific to Epson compatibles, although we are pleased to help in solution of editing difficulties.

PRINT-FILER (C) 1989 RAMBIT

BaPAUG PD Library Additions

- ST ST ST ST ST ST ST ST ST ST ST ST
ST1: ANI-ST £2.50
Originally sold as Aegis Animator for £80, Ani-ST must now be the ultimate PD bargain. Disk includes documentation and examples.
8 Bit 8 Bit 8 Bit 8 Bit 8 Bit 8
Disk 12: Device Handlers £2.50
Several device handlers including a 80 column display, 80 column lister for 1020 printer, multi-mouse and a null handler. All handlers come complete with documentation and source code.
Disk 13: Just For Fun £2.50
A collection of fun games and demo's including FUNFACE a really nice graphics/music demo from Germany.
Disk 20: Yet Another 80-Column Device Handler (Double Sided) £3.50
This disk contains Simon Trew's 80: device; a complete replacement for the E: device handler. Disk contains full documentation, source code, character editor and some bonus programs.

Send cheque / postal order to BaPAUG, 248 Wimbourne Road, Oakdale, Poole, Dorset BH15 3EF

*** ATARI XL/XE OWNERS ***

Public Domain and shareware software available only from me:

- PD100 - Long digital music demos. Long edited sampled music demo and speech files. Price: £2.00
PD102 - Music Machine. The P.D. forerunner of D.M.S. (commercial program available from me), with limited samples and limited editor program. Price: £1.50
PD103 - The "this and that" disk. Music, new versions of DOS, a maze game and lots more! Price: £2.00
SW104 - D.D.S. A new operating environment written in Turbo BASIC (copy on disk), with pop-up calculator, real-time clock, database, diary and a new version of DOS. Price: £5.00, including shareware registration fee.

Other disks also available. For full catalogue, send a SAE, or to order send a cheque / P.O. to:

DEAN GARRAGHTY, 62 THOMSON AVENUE, BALBY, DONCASTER, DN4 0NU

The 1989 BaPAUG Awards

This time last year we caused quite a stir in a couple of national magazines by presenting *The 1988 BaPAUG Awards*. It was meant as a light hearted look at the computer industry and was done on the spur of the moment at one of our meetings. Being amused by this publicity and being sound of limb and mind, and all that junk, we have decided to present *The 1989 BaPAUG Awards*.

We have not repeated all the categories as last time because, to be quite honest, we couldn't think of winners for them all, though we have made up for this by presenting some new ones. For each category, last years winner (if there was one) is shown within brackets.

Best 8-bit Software

Footballer Of The Year by Gremlin
(Draconus by Zeppelin)

Best 8-bit Software House

Zeppelin
(Zeppelin)

Best ST Software

Kick Off by Anco
(Falcon by Mirrorsoft)

Best ST Software House

Mandarin
(Mirrorsoft)

Worst ST Software

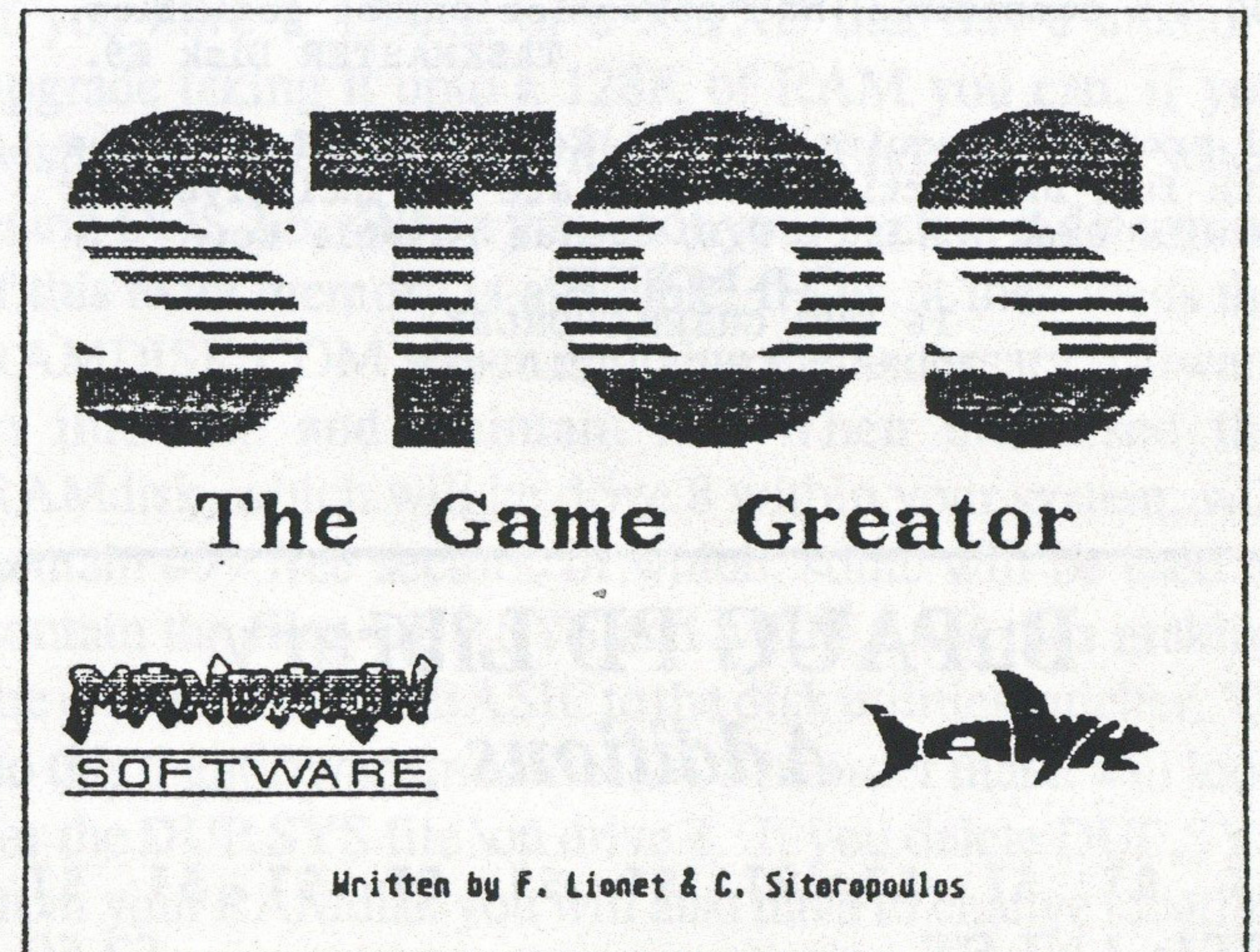
1st BASIC by HiSoft
(Soccer Supremo by Cryssus)

Best Magazine

ST Format
(Page 6)

Worst Magazine

Popular Computer Weekly
(New Computer Express)



8 Bit Software Roundup

Atari 8-Bit owners are in for a real treat from now on, with software houses old and new releasing new titles, re-releasing old titles, and even converting games previously un-released on the Atari. Look out for the following:

Alternative: Pro Mountain Bike, Spitfire 40, Strike Force Harrier, Trailblazer, Who Dares Wins 2, Postman Pat, Fruit Machine Simulator.

Atari: Nucleus, Black lamp, Cygnus X1, Tiger attack, Speedhawk, Heartache, Slingshot, Super Soccer, Contagion, Z-Force, Zone Patrol, Tube Baddies.

Atlantis: Crack Up, Gun Fighter, Encounter.

B. Bytes: Zoltan Escape, Escaper, A-zone, Five dice, Darg, Cyborstein, B. Base 1.

Byte Back: Cosmic Pirate, Screaming Wings, Domain of the Undead, Little devil, Mad Jax, Speed Run, Kendo Warrior.

CDS: System 8

Harlequin: Menace, Last Ninja 2, Paperboy, Fiendish Freddie, Ghosts and Goblins, 1942.

Kixx: Gauntlet, Ace of Aces, Footballer of the Year, Hardball.

Level 9: Scapeghost (their last text only adventure and possibly their last Atari 8 bit release).

Red Rat: Hawkquest. (An instant classic, BUY IT).

Ricochet: Ballblazer.

Tynesoft: The Last Guardian.

Zeppelin: Zybex, Speed Ace, Draconus, Ninja Commando, Phantom, Mirax Force, Sidewinder, Living Daylights, Kenny Dalglish Soccer Manager, Go Cart Simulator.

Who said there was no Atari 8-BIT software being written these days. Keep up the good work lads.

Ian Brooker

Best 8-bit PD

The Big Demo

Best ST PD

Ani-ST

Best After Sales Backup

Columbia Computers

(Columbia Computers)

Worst After Sales Backup

Atari UK

(Comets)

Best Computer Show

Alternative Micro Shows

London & Stafford

Worst Computer Show

The PC Show; Earls Court

Best 8-bit Serious Software

Supplier

Frontier Software

(Computer House)

Best ST Serious Software Supplier

GFA

(HiSoft)

Best Programmers

The Bitmap Brothers

(The Bitmap Brothers)

Best Wargame Simulation

Borodino by ARC

Biggest Disappointment

Atari at Atari Shows

Hardware Flop of the Year

Amstrad PC2386

(Sinclair PC200)

Hardware Success of the Year

Atari Portfolio

Best ST Serious Software

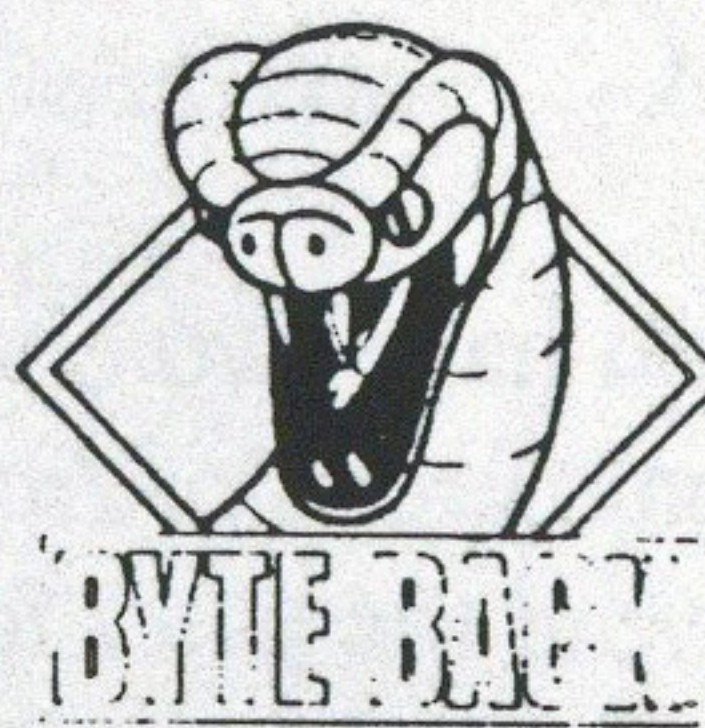
STOS by Mandarin

Best 8-bit Serious Software

SpartaDOS X by ICD

8 Bit Review

MAD JAX
BYTE BACK SOFTWARE
£2.99 CASSETTE (64K)



Civilisation has been devastated by nuclear war and everybody now lives in domed cities, in areas where radiation was at tolerable levels. Your dome is under attack from another city and you have been chosen to take your new, heavily armoured killing machine out onto the streets to find out who is attacking, and to neutralise the threat. So off you trundle in your buggy down the city streets, shooting anything that moves (they are all hostile), guzzling fuel at an alarming rate, but don't worry there's plenty more lying around, as well as weapons, to be picked up as you go.

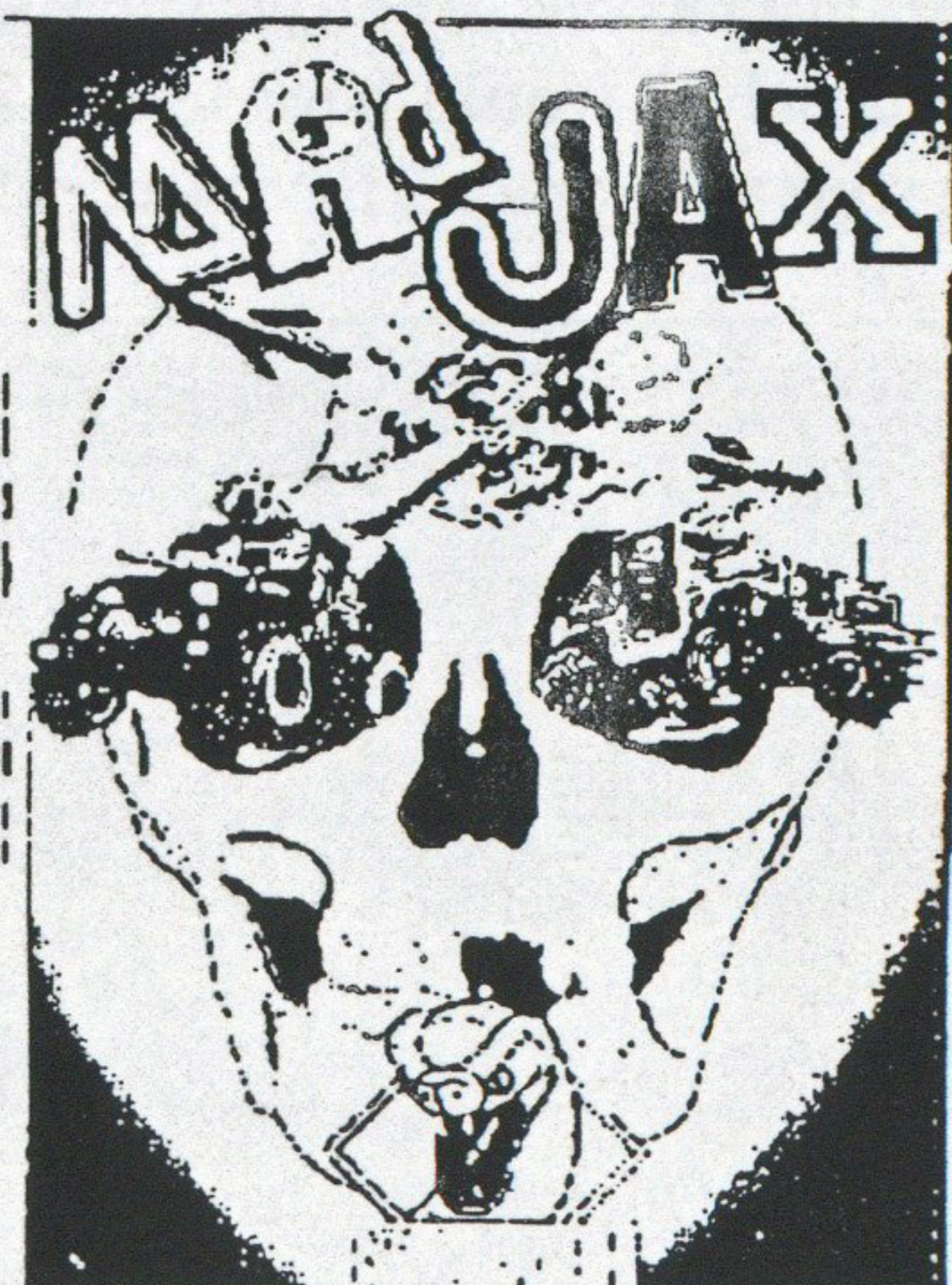
Graphics are well drawn and colourful and scrolling smooth, while unimpressive sound effects are made up for by high quality, catchy background music.

Game play involves steering your buggy through the streets,

shooting all and sundry and picking up stray fuel and weapons. But your enemy moves erratically and is difficult to anticipate, various weapons found along the way will have different attributes, such as faster firing rates, shooting backwards etc. This all adds up to a challenging game that gets you all too easily killed off at first, and whilst pulling you hair out having just one more go (and another, and another). YES its addictive.

All in all this is a very nice, very playable game, with great feet tapping music. At only £2.99 the game is excellent value. BUY IT.

Reviewed by Ian Brooker



Writing A Bulletin Board On The Atari 8 Bit

By James Bastable

1. Introduction

Let me tell you now, running a Bulletin Board is hard work and don't let anyone tell you different. But it's worth it. As a Sysop (SYStem OPERator) you will make many new friends and gain a wealth of knowledge never before dreamed of, believe me. I have been running a Bulletin Board on the Atari 8-Bit for over three years now and have never once looked back (people who know me will tell you different but that was just my off days).

If you have a Atari 8-bit (and the necessary equipment) there is no reason why you cannot run a BBS (Bulletin Board Service) on it, the machine is well suited. Even if you have now upgraded (?) and still have your 8-bit equipment you could run a BBS devoted to your new machine. That way you can leave your new machine free while still having a permanent link with other alike computer users.

This article will attempt to show the more experienced basic programmer how to go about writing a Bulletin Board. Contrary to popular belief, Atari Basic is quite capable of controlling a Bulletin Board of speeds up to 1200 baud which will suit most purposes. For speeds higher, I would recommend the inclusion of Machine Code routines to handle input and output.

Anyway, let's get straight down to it. Before you start thinking about software you must have the correct hardware. Aside from your computer and TV/Monitor you will need a Modem capable of Auto-Answer, a RS232 interface and at least one disk drive.

The computer you use should have at least 48k of memory. All ready-to-run software packages will need this and you will certainly need this amount of memory if you decide to program the software yourself.

1.1 The Modem

I have had experience with three different Modems, the WS2000 with the Auto-Answer board fitted, the Miracom WS4000 and the Pace Linnet. In theory, any Hayes compatible Modem will be suitable although I have witnessed difficulties with the Tandata TM512 which seems to have some strange quirks.

The WS2000 is quite a basic modem (often referred to being a dumb-modem) which effectively runs at two speeds either 300 baud or 1200/75 baud. This particular modem must have the Auto-Answer board installed to work with a Bulletin Board. I originally used this modem for an earlier BBS of mine running at only 300 Baud only, I would have used 1200/75 if I had thought it was possible at the time. It is only recently that this speed has been made

```
100 CLOSE #1 :REM To be sure.
110 OPEN #1,1,0,"RI:" :REM Open to Port
120 XIO 34,#1,0,240,"RI:" :REM Configur
e the port.
130 STATUS #1
140 IF PEEK(747)=253 THEN REM Modem ha
s answered call.
150 GOTO 130:REM Modem not answered, re
try.
```

Listing One: Auto Answer

possible, without speed buffering which the WS2000 is incapable of, on the Atari 8-bit thanks to Geoff McHugh whose article appeared in issue 36 of Page 6. The WS2000 cannot baud rate scan meaning that it cannot automatically detect a incoming speed from a caller and set itself accordingly.

The WS3000 and Pace Linnet are classed as Hayes compatible modems and have Auto-Answer as standard. They can also Baud rate scan which makes them ideal for using on a Bulletin Board. Refer to their manuals for set-up. As I said before, any Hayes compatible should be perfect for running a Bulletin Board.

1.2 The Interface

To the best of knowledge, all RS232 interfaces are suitable for Bulletin Board operation. Again, I can talk from experience with the 850, the P:R: Connection, the Datatari and the one that Derek Fern sells (021-353-5730) in kit form. The Derek Fern and Datatari Interfaces need a special software handler to be loaded before they can be used. This RS232 handler is in the Public Domain and is available on the ARK COMMS DISK #1 in the Page 6 library or it can be downloaded from my Bulletin Board, The City (021-353-5486). Once this handler has been loaded the RS232 interface is treated as a standard 850 Port 1.

1.3 Disk Drives

A modest Bulletin Board can be set up using just one drive. Using a ram-disk such as on the 130XE or the Rambo will boost speed and storage capacity. However, you will find that two or more disk drives will be essential once you start adding Public Domain downloads to your Board. I actually use the SupraDrive 20 megabyte hard disk, though this is no longer available. As a matter of interest, I have recently made contact with a company in America which is selling interfaces for hard drives or complete kits ranging from 5 Megabyte removables to 80 Megabyte fixed. For further information contact Computer Software Services, P. O. Box 17660, Rochester, NY 14617, USA, Phone 716-586-5545

1.4 Software

You have two choices. A) Use a Bulletin Board program either commercial or from the Public Domain or B) write your own. I favour the latter for reasons I will explain later.

In the Public Domain I know of two programs that allow you to run a Bulletin Board. The first is ECABBS which is very basic and FoRem which was released in many different versions of complexity. Documentation for these two is sparse to say the least and for this reason I would not recommend them for the beginner. Having said that most Atari Bulletin Boards in this country were run using FoRem with a certain degree of success.

On the commercial front, I know of only one BB program, Bulletin Board Construction Set (BBCS for short) from the Antic catalogue (0101 415 957-0886). As far as I know, this program is still available. BBCS, as with all Antic programs, comes complete with clear and precise instructions. The program is very powerful with some nice features but does have a tendency to crash now and then. For the casual Bulletin Board operator, this program is ideal

but, because of the crashing problems, the serious operator will find it unreliable. If a choice had to be made, I would recommend BBBS over ECABBS and FoRem every time.

This short overview of BB programs explains why I decided to write my own software. At first sight writing your own BB software seems a daunting task but let me assure you this is not the case although Basic programming experience is essential. If you are a Machine Code programmer you should have the experience to program your software with little difficulty, this article is aimed at the experienced Basic programmer who thinks that writing a BB program is too complex (which it's not) or does not have access to the appropriate information.

The major set back for most programmers is the mistaken belief that programming the 850 interface (or anything that emulates it) is beyond their scope. I will present to you Basic routines that I know work and it is not my intention for this article to be a substitute for the 850 Interface Manual. If you can get hold of or already own the 850 Manual, I suggest you use this article to compliment it.

My own Bulletin Board runs purely in Basic without a Machine Code routine in sight (except one which really isn't necessary but is there just to speed up a certain operation). However, I do use Basic XE for ease of programming but nothing else. Turbo Basic can be used for the same reason and, of course, an increase in speed.

Writing a BB program can be quite arduous but not in areas which prevent most from making the attempt. The program itself must accomplish three tasks:

- 1) Handling input and output from the Modem. The easy part.
- 2) Control a database of messages. Can be hard or easy depending on how many messages you want to store.
- 3) Be user friendly so as to allow the user to log on, do what he has to do and log-off in the minimum amount of time.

The above functions should be transparent to a caller but what must be apparent is an interesting Bulletin Board. It is easy to be so involved with the programming aspect that you lose sight of the important part, the board must be interesting. There is nothing worse than going to all the trouble of creating a Board only to find no-one calls because you have not put enough effort into the layout and content of the Board.

Just to wet your appetite (and to keep you interested) let me show you how easy it is to use Atari Basic with a modem. Listing One shows you how to detect when any modem has answered an incoming call. There, that was easy. If PEEKing location 747 had equalled 253 then we know that someone has rung, the modem has answered and a successful connection has been made. To find this out I used simple trial and error but when you find something that works, you stick with it. It is not important to know how the XIO command works, the fact that it does should be enough. However, I will go into greater detail later.

2. RS232 XIO Commands

XIO commands may seem, on the surface, to be complex but are very easy to use. The fact that one XIO command can perform many functions often leads the programmer into believing that XIO usage is confusing. I will deal with each necessary XIO command in some depth and my lengthy explanations may appear to contradict my earlier statement. I can only repeat that, once studied, the XIO commands are very straightforward.

To illustrate this article I have put together a short program (Listing Two) which contains the necessary XIO and OPEN commands you will need to write your Bulletin Board. There is no need to type the program in if you do not wish to as it does nothing spectacular. It is important, however, for you study how each line of the program works.

Let's go through the program line by line:

Line 90 CLOSEs the channel which is a very good habit to get into especially when using the 850 interface, as you will see later.

Line 100 OPENs channel 1 to Port 1 of the 850 in Read only mode (similar to the disk OPEN). It could have been set to Write only (OPEN #1,8,0,"R1:") but not to read/write (OPEN #1,13,0,"R1:") as the following line would not work.

Line 110 is XIO command 34 which switches the DTR (Data Terminal Ready) and RTS (Request to Send) lines on. Unless this XIO command is issued a lot of modems on the market will not enter into Auto Answer mode. The number 240 in Aux1 in the XIO command sets this up. Again, this command is explained in greater detail later.

Line 120 issues the STATUS command, the X variable being a dummy. When STATUS is issued, information about the condition of the port is updated into memory location 747.

Line 130 PEEKs location 747 and checks whether it



```

10 REM A simple example of input &
20 REM output to the RS232 interface.
30 DIM ANS$(40),BUF$(500)
40 REM -----
50 REM -This routine detects if the-
60 REM -Modem has made a successful-
70 REM -connection.
80 REM -----
90 CLOSE #1 REM To be sure.
100 OPEN #1,4,0,"R1":REM Port 1
110 XIO 34,#1,160,0,"R1:"
120 STATUS #1,X
130 IF PEEK(747)=253 THEN 210
140 GOTO 120
150 REM -----
160 REM -If the program reaches here-
170 REM -the Modem has answered. Now-
180 REM -configure port 1 to enable -
190 REM -input and output.
200 REM -----
210 CLOSE #1:REM To be sure.
220 XIO 36,#1,0,7,"R1":REM Set to 300
    Baud.
230 OPEN #1,13,0,"R1":REM Open Port 1
    or input & output.
240 XIO 38,#1,64,0,"R1":REM Tells RS2
    32 to send line feeds.
250 XIO 40,#1,0,0,"R1":REM Start Conc
    urrent Mode.
260 REM -----
270 REM -Now we can send some data -
280 REM -to the RS232. We'll ask for-
290 REM -the name of the caller.
300 REM -----
310 BUF$="What is your name caller?"
320 PRINT #1,BUF$:REM Send Data
330 REM -----
340 REM -Now we'll check for data -
350 REM -incoming
360 REM -----
370 POKE 747,0:REM Clear the memory
380 POKE 748,0:REM locations to check
390 CNT=1:ANS$=""
400 STATUS #1,X
410 IF PEEK(747)=0 THEN 400
420 REM Line 430 waits until data data
    is in buffer.
430 GET #1,X:REM Pull one byte from bu
    ffer
440 IF X=155 THEN GOTO 520:REM 155=ret
    urn key pressed.
450 ANS$(CNT,CNT)=CHR$(X):REM Store da
    ta in string
460 CNT=CNT+1:GOTO 400
470 REM -----
480 REM -Should now have a name in -
490 REM -in ans$. Now we will send -
500 REM -a welcome message.
510 REM -----
520 BUF$="Welcome ":BUF$(LEN(BUF$)+1)=
    ANS$
530 BUF$(LEN(BUF$)+1)=" to my Bulletin
    "
540 PRINT #1,BUF$
550 BUF$="Board. As this is only an ex
    ample."
560 PRINT #1,BUF$
570 BUF$="I will log you off now. Good
    bye."
580 PRINT #1,BUF$
590 PRINT #1:REM sends a carriage retu
    rn only.
600 REM -----
610 REM -Now we will force the modem-
620 REM -to drop the call.
630 REM -----
640 CLOSE #1
650 OPEN #1,8,0,"R1:"
660 XIO 34,#1,160,0,"R1":REM this line
    does the job.
670 RUN

```

Listing Two: RS232 I/O Example

is equal to 253. If it does then the modem has made a successful connection. If not the program loops back to line 120.

Line 210 again the channel is closed because the port is going to be re-configured.

Line 220 is the XIO 36 command. This takes a little bit more explaining. All XIO commands have two auxiliaries in this form XIO COM,#CN,AUX1,AUX2,"Rx:". Where COM equals the XIO number in this case 36, CN equals the channel number, AUX1 and AUX2 equal auxiliary 1 and 2 respectively. In the case of XIO 34 auxiliary 2 did nothing and can do nothing. With XIO 36, both AUX1 and AUX2 are important to configuring the RS232 port.

Let's look at AUX1 of XIO 36 first. The value placed here tells the RS232 port what baud rate to work at. Here is a list of the corresponding values:-

VALUE	BAUD RATE	VALUE	BAUD RATE
0	300	1	45.5
2	50	3	56.875
4	75	5	110
6	134.5	7	150
8	300	9	600
10	1200	11	1800
12	2400	13	4800
14	9600	15	9600

So a zero (or 8 in this case only) in AUX1 will configure the port to 300 Baud. To this AUX1 a second value can be added which tells the port the word size (the amount of bits in a single character) that will be used. You need not worry about this as 99% of Bulletin Boards use 8-bit word length.

Value	Word Size (Bits)
0	8
16	7
32	6
48	5

So you would add the value from this table to the value for the baud rate you wish to work at. Now to this value you can add yet another to tell the port how many stop bits must be sent with each word (99.999 % of the time it's 1 stop bit).

VALUE	STOP BITS
0	1
128	2

In the program, I set the baud rate to 300 (value 0), the word size to 8 bits (value 0) and the stop bits to 1 (value 0). 0+0+0=0.

The second auxiliary of XIO 36 specifies which lines are to be monitored. The following chart indicates which value needs to be placed in AUX2:

Value	Lines Monitored
0	NONE
1	CRX
2	CTS
3	CTS CRX
4	DSR
5	DSR CRX
6	DSR CTS
7	DSR CTS CRX

Where CRX equals Carrier Detect, DSR equals Data Set Ready and CTS equals Clear To Send. In the program I have opted to monitor all three. Why? Simple, by doing this whenever someone

drops carrier (disconnects unexpectedly) an error is produced which can be trapped in Basic. Very handy when you think about it which I hope you do.

Once an XIO 36 has been issued to a port it remembers it until you switch off the computer or re-configure the port again.

Line 230 OPENS the port for input and output which is what we need for sending and receiving data during the normal running of the Bulletin Board.

Line 240 uses the XIO 38 command which will tell the RS232 what type of ASCII/ATASCII conversion you want, if any, and whether the RS232 will append line feeds. Both AUX1 and AUX2 are used. Let's look at AUX1 first.

The value placed here tells the RS232 whether you want NO TRANSLATION (as in file transfer or if your Bulletin Board is going to work in ATASCII mode), LIGHT TRANSLATION which is the normal setting for standard ASCII communication or HEAVY TRANSLATION which I never use because it tends to disallow some characters that may be needed. Here is a list of values and their meanings:

Value	Translation
0	Light
16	Heavy
32	No Translation

Both Heavy and Light translation will convert ATASCII carriage return (155) to ASCII (13). They will also ignore the 8th (most significant) bit as all ASCII characters only use 7 bit. NO TRANSLATION means exactly what it says.

To this value we can add another which sets the input Parity mode, thus;

Value	Effect On Input Parity
0	Ignore & do not change
4	Check for odd parity and clear
8	Check for even parity and clear
12	Ignore and clear

The first option (Value 0) is the most common. To effect the Output Parity you can add one of the following values.

Value	Effect On output Parity
0	Do not change
1	Set to odd
2	Set to even
3	Set to one

Once again, the first option (value 0) is most common.

AUX1 of XIO 38 can also instruct the RS232 whether to send a Line Feed character (ASCII 10) after a carriage return or not. Atari 8-bits don't need this but there are some computers that do, so really it should be sent. Most computers that don't need the character appended ignore it anyway so it does no harm.

So, having taken two values from the three tables and added them together, you can now add to this new value a 64 if you wish Line Feeds to be sent or 0 if not.

AUX2 of XIO 38 is only used when HEAVY TRANSLATION is set. In this mode whenever the RS232 ports receives data which cannot be converted to a recognised ASCII character the RS232 will change it to a WON'T TRANSLATE CHARACTER. This character is set by placing a value in AUX2. On the very rare occasions that I have used this mode, I have placed a value of 126 (ATASCII Back-space) in this auxiliary but any value between 0 and 255 can be placed here.

Line 250 configures the port to concurrent mode. AUX1 and

AUX2 serve no purpose so are set to zero. Once concurrent mode has been made active by XIO 40 the break key is disabled which is helpful as long as you are confident you won't need it. You can always press the Reset key if you get into trouble but this can sometimes play havoc with the port configuration. Annoyingly, you cannot access any other external device such as disk drives, cassette players, printers, etc while the program is concurrent mode. There is no way round this, I've tried. Remember me saying that it was important to get into the habit of closing a channel before using OPEN?

Line 310 loads some text into the string BUF\$. This isn't necessary in this case, the text could have just been PRINTed out to the port without being entered into the string first. When you come to actually programming a Bulletin Board you will be reading text from the disk drive and, as you cannot have concurrent mode active while reading from the drive, you will be pushing text into a string anyway.

Line 320 PRINTs the contents of BUF\$ to the RS232. Using the trailing semi-colon to suppress the carriage return from being sent. This is not echoed onto the computer screen which in the real world would be so. A simple PRINT BUF\$; placed here would do the job nicely.

Line 370 and 380 clears the values in locations 747 and 748. These locations now have a different meaning when in concurrent mode. Remember location 747 was used earlier for detecting a successful modem connection. I have not found a use for location 748 but the 850 manual says clear it, so I clear it.

Line 390. CNT is set to one, which is used to keep track of the amount of data that has been received and ANSS\$ is set to a null string where the incoming data will be placed.

Line 400 issues STATUS which updates location 747.

Line 410 PEEKs location 747. Why? I wondered when you was going to ask! In concurrent mode location 747 contains the amount of bytes in the buffer, so by checking if it is greater than zero we know whether data is coming into the RS232 port. As a matter of interest the standard buffer size for the RS232 handler is 32 bytes and if you don't get the data out quick enough you start to lose it. More about that later.

Line 430 GETs a byte from the buffer which must be available otherwise the program would not have reached this far.

Line 440 checks to see whether the value got in line 430 is 155 which is ATASCII carriage return. If the value is 155 then the program knows that the caller has finished typing.

I know what you're thinking! You're saying to yourself, all very well checking for a 155 but what if someone calls who is not using a Atari and they send a ASCII 13 which is a standard code for Carriage Return. I say, aha, but the RS232 interface being a clever little devil knows this and converts it to a 155, so there. It also converts it back when sending. This option was set in line 240 using XIO 38.

Line 450 places the data got in line 430 into the string BUF\$.

Line 460 increments variable CNT which is done every time Data is received other than ATASCII 155. Then loops back to get more data if possible.

Lines 520 to 590 simply loads text into BUF\$ and PRINTs it out to channel 1. This time no trailing semi-colon which means that a Carriage Return is sent (and a Line Feed if a 64 was added to AUX1 of XIO 38).

Line 640 CLOSEs the channel as it is going to altered. As a matter of great interest, whenever the RS232 port is CLOSEd all the contents of the buffer is wiped. I sometimes close a port whenever large amount of data corruption has been detected just to clear out the rubbish..

Line 650 OPENS it in Write only mode. This is used because the following XIO command cannot be used in concurrent mode.

Line 660 turns off RTS and DTR control which forces the modem to drop carrier. This is the opposite to the XIO 34 used in line 110. To calculate these values used here is yet another table:

Value	DTR Control	Value	RTS Control
128	Turn DTR off	32	Turn RTS off
192	Turn DTR on	48	Turn RTS on

So you see, by adding two values together, 128 and 32 equalling 160, both DTR and RTS controls are turned off.

Confused? Don't worry about it, so was I at first but I soon learnt that a lot of the settings are unnecessary when operating a standard Bulletin Board. If you are planning on using RS232 for more specialised applications then I would suggest that you study the matter more by obtaining the excellent manual for the 850 interface but general communications is nothing out of the ordinary so does not require you know everything there is to know.


3. Turbo BASIC and the RS232 Handler

If you intend to use Turbo BASIC to write your Bulletin board software, a word of warning. When TB is loaded it wipes the 850 RS232 handler thus preventing the use of the RS232 ports. Listing 3 will create a file on Drive 1 called 850DRIVE.COM which can be BRUNed (BRUN "D:850DRIVE.COM") to load the 850 after Turbo has loaded. Thank you Arthur Edwards of Powys wherever you are these days.

4. Improving The Data Transfer Rate

Now onto the nitty gritty of writing your own Bulletin Board. By now you should have a grasp of how to use XIO commands, so we'll go onto the problem of speeding up Data transfer rate as which basic struggles with. The problem lies in the PUT and GET commands which, although being ideal for handling individual bytes to and from a device, are just too slow for speeds of 300 baud and above.

Actually, we need not worry too much about incoming data as callers to your Bulletin Board will be sending data at typing speed. This is well below 300 baud as you can imagine. The GET command can handle this happily and still have leave time to echo the characters.

You must remember that any character sent to your Bulletin Board should be echoed two ways. Firstly, it has to be echoed to the caller which is standard Full Duplex operation and, secondly, it should be echoed to your own (local) screen so that you may monitor what is taking place. This second operation may be ignored if speed is essential. So a typical GET 

```

1990 OPEN #1,8,0,"D:850DRIVE.COM"
2000 FOR D=0 TO 87:READ BYTE:PUT #1,BY
TE:NEXT D:CLOSE #1
2010 DATA 255,255,0,56,75,56,169,80,14
1,0,3,169,1,141,1,3,169,63,141,2,3
2011 DATA 169,64,141,3,3,169,5,141,6,3
,141,5,3,169,0,141,4,3,141,9,3
2012 DATA 141,10,3,141,11,3,169,12,141
,8,3,32,89,228,16,1,96,162,11,189,0
2013 DATA 5,157,0,3,202,16,247,32,89,2
28,48,6,32,6,5,108,12,0,96,226,2
2014 DATA 227,2,0,56

```

Listing Three: Creates 850DRIVE.COM to load RS232 handler into Turbo Basic by Arthur Edwards.


```

10 REM assuming that BUF$ holds data.
20 REM assuming that channel 1 is open
   to RS232.
50 REM BAUD holds the value of the Baud
   Rate
60 REM using 0=300 Baud 10=1200 baud.
70 N=BAUD*4
100 IF LEN(BUF$)<N THEN PRINT *1;BUF$;
PRINT BUF$;RETURN
110 Z=INT(LEN(BUF$)/N):CNT=0
120 FOR LP=1 TO Z
130 T$=BUF$(CNT+1,CNT+N)
140 PRINT *1;T$;:PRINT T$;
150 CNT=CNT+N
160 REM Program goes away and checks f
   or incoming data.
170 NEXT LP
180 IF LEN(BUF$)-(Z*N)>0 THEN T$=BUF$(
(Z*N)+1,LEN(BUF$)):PRINT *1;T$;:PRINT
T$;
190 PRINT *1;PRINT
200 T$="":RETURN

```

Listing Four: Text print routine

operation should follow this format:-

```

100 GET *1,CHAR :REM GET FROM MODEM ONE BYTE
110 PUT *1,CHAR :REM ECHO TO CALLERS SCREEN
120 PRINT CHR$(CHAR);:REM ECHO TO LOCAL SCREEN

```

Basic can output data easily in excess of 1200 baud by using the PRINT command. The problem lies in your own screen display (usually referred to as "Local"). If you were to PRINT out a string containing 1500 characters you would either have to print it to your screen first which will cause a delay or after with same effect. Also, think what would happen if a caller wanted to abort a stream of text that is being transmitted?

The answer is to break the string down into smaller parts then PRINT them out to caller. After each PRINT command the program could go away and check whether anything has been typed and whether what has been typed indicates that the caller wishes to pause the listing or abort it completely. During this interval, you could also check whether the caller is still on line by CLOSEing the port or, if using a Hayes modem set correctly, you could check for the character 3 (ASCII 51) which is the Hayes code for 'No Carrier' and then CLOSE the port to check whether the caller is actually still there or not. Listing Four shows the routine I use; but maybe you can think of a better one.

Note that even though this is a subroutine the line numbers are quite low. With standard Atari Basic, routines that are the nearer the start of the program are accessed faster than those that are toward the end. This routine will PRINT larger chunks of data out at 1200 baud than at 300 baud.

Receiving data, as I mentioned before, is not that difficult as it is usually typed to a Bulletin Board. With standard Atari Basic you cannot use the command INPUT. Although INPUT is fast enough, it will not stop trying to receive data until it gets a EOL (ATASCII 155) character. This may not, at first seem to be a problem until you start thinking about what would happen if half way through receiving data the carrier was lost (the caller dropped the line). Your program would sit there forever waiting for the ATASCII 155 and what about when you are transferring programs where the incoming data may not contain a 155 at all.

In most circumstances you can simply use the GET command but when you want receive data at speed you are going to need a more efficient command. Fortunately for users of Basic XL, Basic XE and Turbo Basic this command is readily available. If you are using Standard Basic, nothing to be ashamed of, Listing Five

contains a routine that loads a Machine Code program into a string called ML\$ which will duplicate the Binary Get (BGET) found in other Basics. The Binary get is used in this manner:

```
X=USR(ADR(ML$),CHAN,ADR(BUF$),AMOUNT)
```

ADR(ML\$) is the address of Machine Code routine.

CHAN is the Channel Number being accessed.

ADR(BUF\$) is the address of the string you wish the data to be entered into.

AMOUNT is the amount of Bytes you wish to be transferred.

As a bonus the amount of bytes that was actually transferred by the routine is returned into X.

Obviously, the Binary Get can be used with other devices, such as disk drives, for speedily reading in Data.

To ascertain how many bytes of data are needed to be transferred from the RS232 and entered into the USR parameter AMOUNT, you will need to do the following:

When in concurrent mode (XIO 40) issue a STATUS and PEEK memory location 747 which will tell your program how many bytes are in the buffer. If there is none, then your program could go away and check whether the caller is still on line. The value found in location 747 is placed into variable AMOUNT.

As a recap, if you have set up port 1 correctly using XIO 36, simply OPENing and CLOSEing the channel will tell you whether the caller is still there. If the caller has dropped carrier this action will cause a TRAPable error.

When you are receiving information on a Bulletin Board always assume that the caller is going to do something stupid. For example, typing in lower case when HIGHER case is required. Your program should allow for this and alter the data accordingly.

Also, try to remember that your caller may suffer with line noise (data corruption by the telephone line) so check for characters that are not ever used for typing messages or commands and filter them out. To find these look through a list of ASCII characters and codes.

5. XMODEM File Transfer

Xmodem is a standard protocol used in most communications programs for the Uploading and Downloading of files, be they text or program data. Because the telephone line has a tendency to corrupt data, such a protocol is essential. Xmodem is very simple to use and to program into a Bulletin Board.

There are many variations of Xmodem, all boast different features and different degrees of reliability. I have used Standard Xmodem for over eight years now and can honestly say that I have never had a failure that was not detected by the protocol.

The Xmodem protocol is simplicity itself although it can never be said to be the fastest way of transferring data.

When sending or receiving using Xmodem, the data is split up into blocks of 128 bytes. Each block is transferred one at a time along with three bytes on the front end to signify the start of a block and the number of the block being sent. At the end of a block a one byte calculated Checksum figure is sent which is used to check the accuracy of the data transferred.

A typical block of data will look like this:-

```
<SOH><BLOCK><255-BLOCK><128 BYTES OF DATA><CHECKSUM>
```

A SOH is ASCII character 1 which is always sent at the very beginning of a Block. The BLOCK is the actual number of the Block sent. So if this was the first this would equal 1, the second

would equal 2 and so on. When the BLOCK number reaches 255 the value is wrapped round to 0 and not to a 1 as you might expect. The third byte (255-BLOCK) is sent to confirm the Block number (BLOCK) previously sent. Following on from these three bytes, the 128 bytes of actual raw data is sent. The last byte to be sent is the CHECKSUM which is calculated by adding together ALL the previous bytes sent and the low byte of this value is calculated as the checksum. This can be expressed as a formula.

```
CHECKSUM=X-INT(X/256)*256
```

Where X equals the total of all the previous bytes added together including the SOH, BLOCK and the 255-BLOCK. As a matter of interest Atari Basic can calculate this Checksum far more easily, like this:

```
CHECKSUM=ASC(CHR$(X))
```

Which will place the low byte of X (total bytes added together) into the variable CHECKSUM.

There are many other factors we have to consider when transferring data. The best way to is to discuss a sample program line by line and highlight the areas that need expanding (see Listing Seven). Once again it is not necessary to type in the program as it only serves as an example.

Line 70 Sets the values of the variable that are to be used. BLK stores the Block Number being sent. TRY records the amount of attempts that will be made sending a Block before aborting. NAK holds the ASCII value 21 which is used in two ways; one to indicate that the program is ready to receive data and one to indicate that the Block received was not correct. ACK holds the ASCII value 6 which is used to indicate that the Block received is correct. CAN holds the ASCII value 6 which is used when a transfer is to be aborted.

Line 80 sets more variables. SOH is set to ASCII 1 which is always sent at the beginning of a Block. CHK is set to 0 which will be used to store the yet to be calculated CHECKSUM value sent at the end of a Block. FLAG is set to NAK and will be used to indicate the condition of the transfer to the program. A\$ is DIMensioned to 128 which is the amount of Data that will be received in any one Block. BUF\$ is used to store the correct Data once it has been received.

Line 90 is the start of the transfer program Loop and will always check the amount of attempts that have been made when transferring a Block of data. Standard Xmodem will always abort if a Block data is received or sent more than 10 times. If the Block is incorrectly received more than 10 times, FLAG is set to CAN (ASCII 24) which will tell the Sending computer that too many attempts have been made and that the line must be so bad that Xmodem transfer is unsuitable at present.

Line 100 PUTs a byte to the RS232. The first time round this will be a NAK which is used to indicate that the program is ready to receive. FLAG was set to NAK in line 80. From now on FLAG will either contain a ACK if the Block was received correctly or a NAK if not.

Line 110 checks to see whether FLAG was set to CAN in line 110. If so, the transfer should be aborted.

Line 120 gets the first byte of data being sent. This should obviously be a 1 (SOH). CHK is set to value of the first byte received and FLAG is set to ACK in preparation of receiving the Block correctly.

Line 130 checks to see whether the byte received is a CAN (24) which tells the program whether the data being sent has been aborted or not by the caller. I actually check this twice, when I program Xmodem for my own use, and a lot of software is adopting

this double check. It is possible that Line Noise over the telephone link could corrupt incoming data to the degree that a byte could be changed to a 24 forcing the program to abort unexpectedly. If you do adopt this method you should also send CAN twice when actually aborting. In this case Line 130 would read:

```
IF X=CAN THEN GET *1,X:IF X=CAN THEN REM *ABORTED*
```

Line 140 checks to see whether the first byte received was a 1 (SOH) and if not sets FLAG to a NAK (21) which tell the program that the data received was corrupt. FLAG will eventually be sent out to indicate if the block was received correctly or not.

Line 150 GETs the second byte which should contain the block number. This Value is added to CHK.

Line 160 Checks to see whether the value received corresponds to the programs record of the Block being received and sets FLAG accordingly.

Line 170 GETs the third byte which should contain the Block number minus 1. This value is also added to CHK.



```
10 DIM HL$(46)
20 FOR LP=1 TO 46
30 READ A
40 HL$(LP,LP)=CHR$(A)
50 NEXT LP
100 DATA 32,68,218,104,104,104,10,10,1
0,10,170,169,7,157,66,3,104,157,69,3,1
04,157
110 DATA 68,3,104,157,73,3,104,157,72,
3,32,86,228,189,72,3,133,212,189,73,3,
133,213,96
```

Listing Five: Creates M/C program for Binary Get

```
10 ;-----
20 ;BINARY GET BY JAMES BASTABLE.
30 ;-----
40 ;USE FORMAT
50 ;X=USR(XXXX,CHAN,ADDR,AMOUNT)
60 ;CODE IS RELOCATABLE
70 ;-----
80 ICCON=$0342
90 ICBAR=$0345
0100 ICBAL=$0344
0110 ICBLH=$0349
0120 ICBLL=$0348
0130 CI0V=$E456
0140 *= $0600
0150 JSR $D44 ;CLEAR 212-217
0160 PLA ;BYTES PASSED
0170 PLA ;
0180 PLA ;GET CHANNEL NUMBER
0190 ASL A
0200 ASL A
0210 ASL A
0220 ASL A ;MULTIPLY BY 16
0230 TAX ;TRANSFER TO X
0240 LDA #7 ;DR 11 FOR BPUT
0250 STA ICCON,X
0260 PLA ;GET BUFFER ADDRESS
0270 STA ICBAR,X
0280 PLA
0290 STA ICBAL,X
0300 PLA ;GET BYTE AMOUNT
0310 STA ICBLH,X
0320 PLA
0330 STA ICBLL,X
0340 JSR CI0V ;DO THE JOB
0350 LDA ICBLL,X
0360 STA 212 ;RETURN BYTES
0370 LDA ICBLH,X ;TO BASIC
0380 STA 213
0390 RTS ;ALL DONE
```

Listing Six: Source Code For Binary Get

Line 180 Checks this value and sets FLAG.

Line 190 is the start of the loop which will GET 128 bytes of data.

Line 200 GETs the ASCII value of the data and adds it into the string A\$

Line 210 adds the value to CHK.

Line 220 ends the loop.

Line 230 GETs the last byte of the Block. This value should be the calculated CHECKSUM from the sender.

Line 240 calculates the low byte from CHK and places it in the variable CHECKSUM.

Line 250 compares the Checksum received with the checksum calculated from the Data received. If these values do not agree then it is assumed that the data received was corrupt and FLAG is set to NAK.

Line 260 checks whether FLAG equals a NAK (21) which would mean that some part of the data received was incorrect. If so then TRY is incremented and the program is forced back to line 90 where another attempt is made to receive data.

Line 270. If this line is reached then the data received must be correct. BLK is incremented as the program will now attempt the next Block of data.

Line 280 places the contents of A\$ into BUF\$ where it is stored for transferring to disk later.

Line 290 TRY is set to 1 ready to start receiving a new data block and the program is forced to line 90 to restart the Xmodem loop.

The program should explain the basics of Xmodem transfer but it is up to the programmer to check for other errors that may occur.

The most common data corruption that must be checked for is that of Line Noise causing a Data Block to be shorter than it should be. If you were to use the illustrated program as it stands then a short block would cause the program to hang up. As with any

Bulletin Board program all GET operations must be within a loop which checks (by PEEKing location 747) whether there is any actual data to be got. If, after about a minute, your Xmodem program has not received any data, two things should happen; 1> The RS232 must be checked to see whether the caller has dropped carrier or not and 2>, if the caller is still there, a NAK must be sent to indicate that a short block has been received.

Another point to watch out for is when Line Noise causes too many data bytes to be sent. Although this may not cause a problem when a data block is received correctly, it may effect the following data block. To be sure, I always close the RS232 channel to clear the buffer whenever I receive a bad Data Block. That way the next block is not effected by any corruption overspilled from the previous bad block. Any overflow from a previously received correct block (ie corrupt data added to the end) should not really cause too much concern.

Whenever you are transferring data in this way, you must remember to set your RS232 port to NO TRANSLATION by using the XIO command 38. The last thing you want when using Xmodem (or any protocol for that matter) is to have the RS232 altering the data that is being received or transmitted.

One final point that you may, or may not, have thought about is what to do when the amount of data you want to transmit is not divisible exactly by 128. Obviously, the answer is to add extra bytes to the end. The question is what byte value should be added? I have experimented with many and most seem to cause problems when the file being transmitted is a program (some DOS's object to extra bytes tagged onto a program). In my experience, the safest answer is to check what the value is of the last byte of data to be transmitted and append this value. This method appears to be agreeable to most computer systems, in particular the Atari 8-bit.

As I said before, Standard Xmodem is reliable for most purposes and certainly when transferring data at speeds of 2400 baud and below. On my Bulletin Board I have not installed any other file transfer protocol and have not as yet found it necessary to do otherwise. Maybe, this will change when higher speed modems become more accessible but for the time being Xmodem will do very nicely.

6. Conclusion

That concludes my article to assist you with writing a Bulletin Board. I hope that some of you will at least attempt it as I have gained many months of satisfaction from the challenge. The Atari 8-bit is not supported as well as it could be with Bulletin Boards and, I'm sure, more Bulletin Boards for the 8-bit Atari can only do good.

The subjects that I have not touched on, like the message Database and Menu manipulation, will have to be programmed by you but not necessarily without my help. I can be contacted via my Bulletin Board (The City) on 021-353-5486 at speeds of 300, 1200/75 and 1200/1200 baud (soon to be upgraded to include 2400/2400 and still using Basic). If you would like to see more articles of this nature, let me know and I'll do my best to oblige.

Articles Required

In-order to keep the quality and quantity within 8:16 at its current high standard we need more articles.

If you have an article please send it to BaPAUG, 248 Wimborne Rd., Oakdale, Poole, Dorset.

Remember each article published extends your subscription by one.

```
10 REM -----
20 REM EXAMPLE XMODEM RECEIVE ROUTINE
30 REM ASSUMING THAT CHANNEL #1 IS
40 REM OPEN TO RS232 IN CONCURRENT
50 REM MODE.
60 REM -----
70 BLK=1:TRY=1:NAK=21:ACK=6:CAN=24
80 SDH=1:CHK=0:FLAG=NAK:DIN A$(128),BU
F$(1500)
90 IF TRY>10 THEN FLAG=CAN:REM *ABORTE
D*
100 PUT #1,FLAG
110 IF FLAG=CAN THEN REM *ABORT TRANSF
ER*
120 GET #1,X:CHK=X:FLAG=ACK
130 IF X=CAN THEN REM *ABORTED*
140 IF X<>SDH THEN FLAG=NAK:REM BAD BL
OCK
150 GET #1,X:CHK=CHK+X
160 IF X<>BLK THEN FLAG=NAK:REM BAD BL
OCK
170 GET #1,X:CHK=CHK+X
180 IF X<>255-BLK THEN FLAG=NAK:REM BA
D BLOCK
190 FOR LP=1 TO 128
200 GET #1,X:A$(LP,LP)=CHR$(X)
210 CHK=CHK+X
220 NEXT LP
230 GET #1,X
240 CHECKSUM=ASC(CHR$(CHK))
250 IF X<>CHK THEN FLAG=NAK:REM BAD CH
ECKSUM
260 IF FLAG=NAK THEN TRY=TRY+1:GOTO 90
270 BLK=BLK+1:REM ALL WENT WELL
280 BUF$(LEN(BUF$)+1)=A$
290 TRY=1:GOTO 90
```

Listing Seven: Example XMODEM Receive Routine

User Group File

Looking for your local User Group? Then check out the map below to find the nearest group and give its contact a call.

Name : Birmingham User Group
Contact : Wilf Coton
31 Barn Meadow, Yardley, Birmingham, B25 8YT
Notes : XL-ST, Meetings, BBS

Name : Bournemouth & Poole Atari User Group (BaPAUG)
Contact : Colin Hunt - 0202 677895
248 Wimborne Road, Oakdale, Poole, Dorset.
Meetings : 1st Friday every month at the Kinson Community Centre, Pelhams, Millhams Lane, Kinson.
Newsletter : 8:16

Name : Cardiff Area ST User Group
Contact : Alan Griffiths
48 Charles Street, Cardiff, CF1 4EF
Notes : ST, Meetings

Name : Fast ST Basic Users Group
Contact : Simon Rush
42 York Road, Rayleigh, Essex SS6 8SB
Notes : ST, Newsletter

Name : HMS Drake's Computer Club
Contact : John Godfrey
Education Centre, HMS Drake, Plymouth, Devon
Notes : XL-ST Others, Meetings

Name : Humberside ST User Group
Contact : Bill Galashan
43 Queensmead, Beverley, N. Humberside HU17 8PQ
Notes : Meetings

Name : John Player Computer Club
Contact : Robin Smith
7 Trafalgar Road, Beeston Rylands, Notts., NG9 1LB
Notes : XL-ST-Others, Meetings

Name : London Atari Computer Enthusiast (LACE)
Contact : Glenn Leader
143 Richmond Road, Leytonstone, London E11 4BT

Name : Merseyside Atari Club
Contact : Ken Hall
84 Fitzroy Way, Liverpool L6 1JS
Notes : ST, Meetings, Newsletter & BBS

Name : Norwich User Group
Contact : Ken Ward - 0603 661149
45 Coleburn Road, Lakenham, Norwich NR1 2NZ
Meetings : 1st Sunday every month. Contact Ken for time and place.

Name : The Programmers Advice Line
Contact : 42 York Road, Rayleigh, Essex SS6 8SB
Notes : ST, Newsletter

Name : Slough Atari Users Club
Contact : Colin Corne
3 Wartley Close, Stoke Poges, Bucks. SL3 6NS
Notes : XL-ST, Meetings

Name : Special Interest Group Atari (SIGATARI)
Contact : Glenn Leader
143 Richmond Road, Leytonstone, London E11 4BT

Name : South West ST User Group
Contact : Karl Fuller
289 Fort Austin Avenue, Crownhill, Plymouth, PL6 5ST
Notes : ST, Meetings, Newsletter

Name : Thames Atari Users
Contact : Anthony Cox
48 Stile Hall Gardens, Chiswick, London, W4 3BU
Notes : XL-ST, Meetings

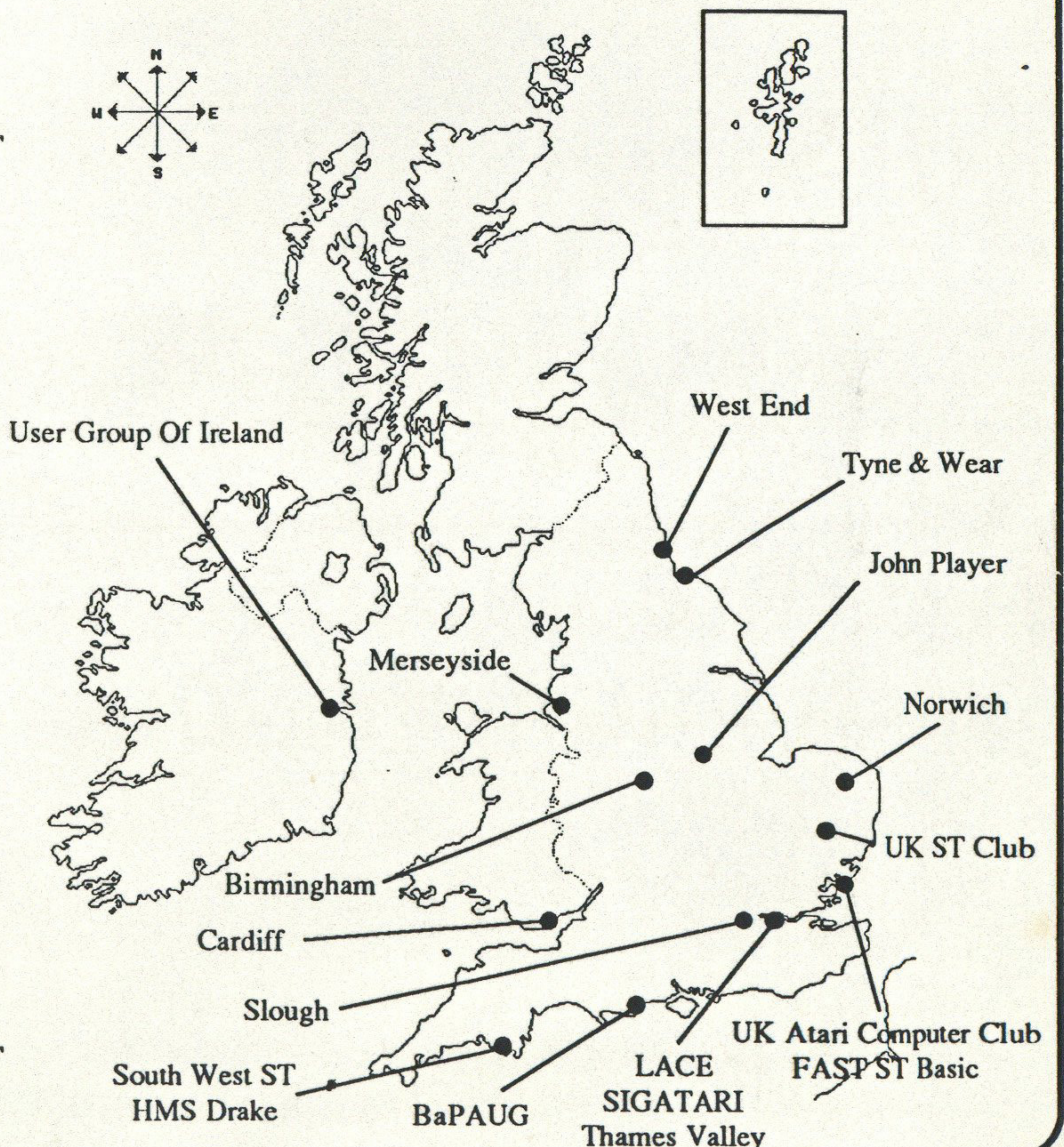
Name : Tyne & Wear Atari User Group
Contact : F. Gregory
20 Maple Avenue, Silkwood, Sunderland SR3 10W
Notes : XL-ST, Meetings, Newsletter

Name : The UK Atari Computer Club
Contact : Roy Smith
PO Box 3, Rayleigh, Essex SS6 2LR
Newsletter : Monitor
Notes : XL-ST, No meetings

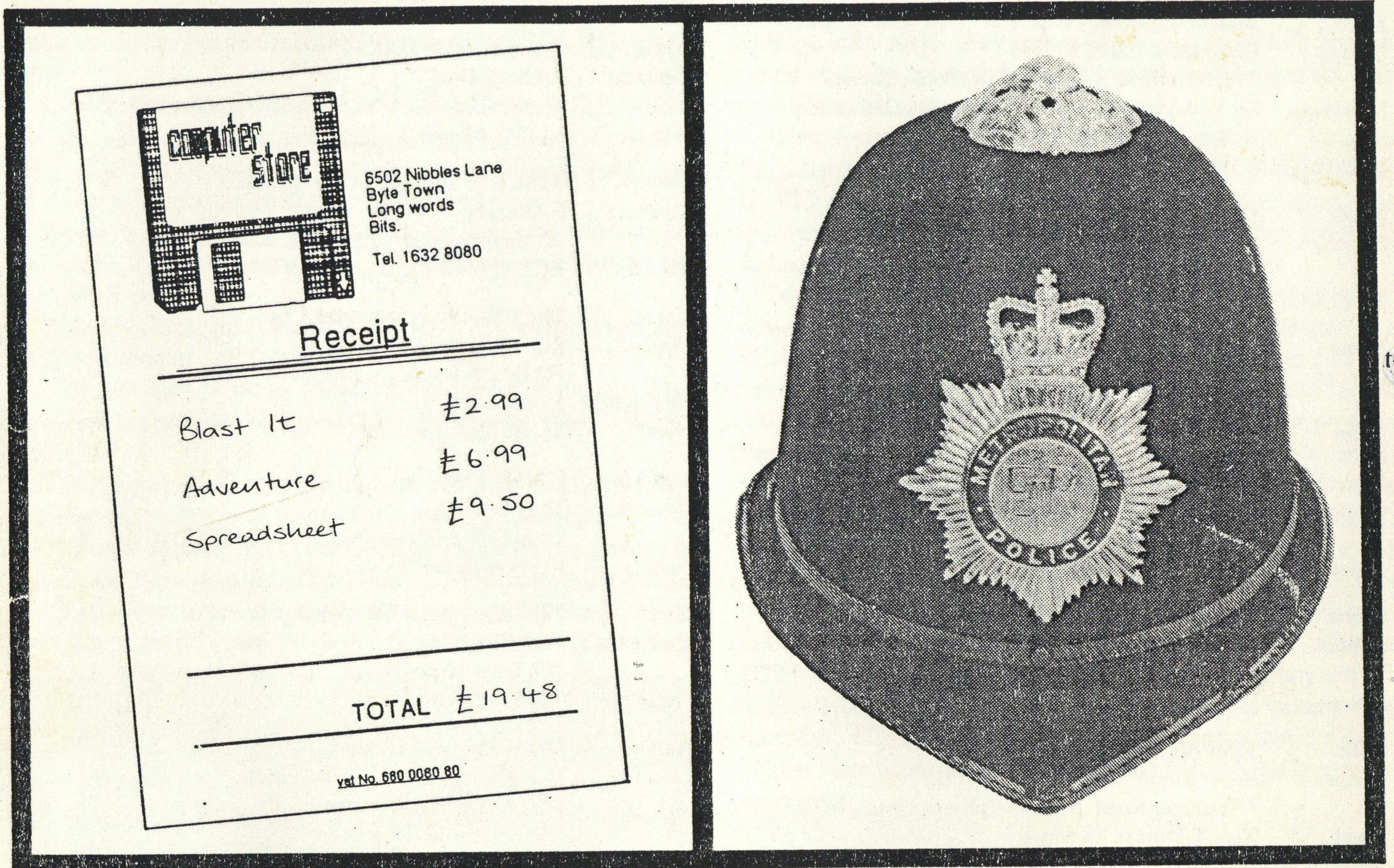
Name : UK ST Club
Contact : Tony Gosling
Swans Nest, Westerfield, Ipswich, IP6 9AJ
Notes : ST, Newsletter

Name : West End Atari User Group
Contact : John Chadwick
208 Janet Street, Byker, Newcastle-Upon-Tyne
Notes : XL-ST, Meetings

Name : User Group Of Ireland
Contact : Mike Casey
3 St. Kevins Park, Kilmacud, Co. Dublin



WARNING



**THIS SORT OF
BILL IF YOU
PAY FOR THEM**

**THIS SORT OF
BILL IF
YOU DON'T**

A pirated game could result in a visit from you know who.

THIS CAMPAIGN IS ORGANISED BY
ELSPA
EUROPEAN LEISURE SOFTWARE
PUBLISHERS ASSOCIATION

*Any information on piracy
should be passed to
The Federation Against Software Theft.
Telephone 01-240 6756*

