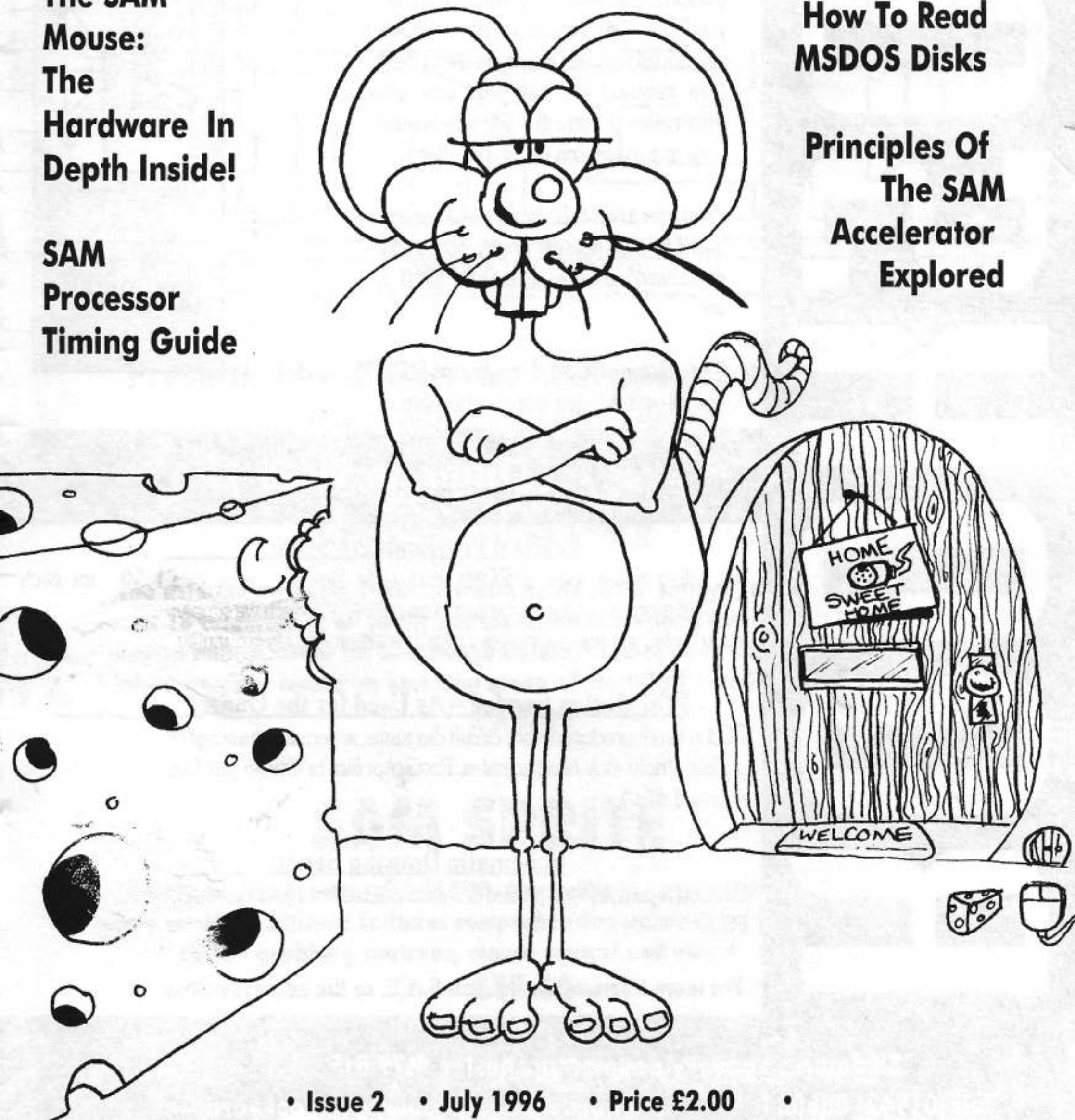
Based On An Idea...

The SAM Mouse: The Hardware In Depth Inside!

SAM Processor Timing Guide

How To Read MSDOS Disks

Principles Of The SAM Accelerator Explored



• Issue 2 • July 1996 • Price £2.00 •

ROOKSOFT

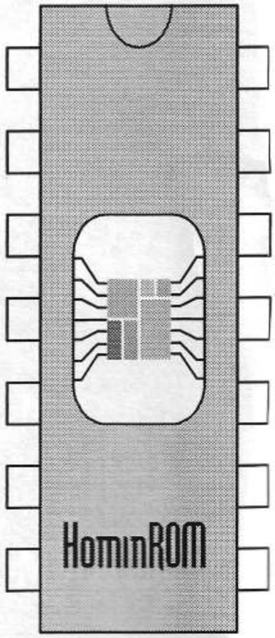
The HominROM

WORRIED ABOUT THE RISK OF THEFT OF YOUR SAM? Own a pre-version 3.0 SAM ROM? Well **ROOKSOFT** have a solution to both of these problems...

The **HominROM** is a v3.0 (equivalent to West Coast v3.1) SAM Coupé ROM, personalised with your own name and postcode on the title screen, making identification much easier in case of theft! (On request we can put any other information in there that you may require – up to a maximum of 32 characters).

This revised ROM also replaces the cassette based function keys with much more useful disk based options such as DIR, BOOT 1, etc.

The **HominROM** costs just **£7.95** and comes with detailed fitting instructions.



Other Services

EPROM Programming Service

We will program most EPROMs from code files. **£1.50** for each EPROM programmed, plus the cost of any EPROMs which we supply; if you prefer, you may supply your own. We will program any quantity.

PCB Design Service – As Used for the Quazar

PCB artwork produced from circuit diagrams, schematic drawings or netlists. Photo etch-resist acetates, PostScript files or Gerber plot files returned. P.O.A.

Schematic Drawing Service

Circuit diagrams drawn, printed & netlists created from your drawings. P.O.A.

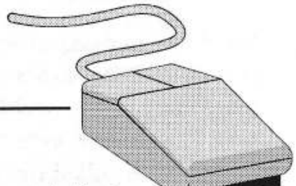
For more information send an S.A.E. to the address below.

38 Squires Lane, Tyldesley, Manchester, M29 8JF
Tel: (01942) 886084 • Cheques made payable to M.Rookyard

Based On An Idea...

Issue 2 – July 1996

CONTENTS



WE HAVE MOVED!
You can now find us at:
38 Squires Lane
Tyldesley
Manchester
M29 8JF

Editorial...

Well, here we are again, slap bang in the middle of another glorious English summer; so while it's absolutely pouring down outside and there's no chance of doing the garden, this seemed as good a time as any to get on with the Editorial. So off we go...

Hello SAM lovers everywhere and welcome to another thrilling episode in the life of Based On An Idea... Doesn't time fly? It seems like only yesterday that we were sitting at the kitchen table struggling over the editorial for Issue 1 and here we are at Issue 2 already – and going from strength to strength. We've had lots of interest from SAM owners across the country (and beyond) and out subscription list is longer than we even dared hope after such a short time.

We'd like to thank everyone who answered the questionnaire handed out with the first issue, because it's only by listening to the people who actually buy and read the magazine that we're able to produce something that everyone will enjoy and find useful. Again we were pleasantly surprised by the response. As promised, five replies were pulled from the hat (by our son Simon, aged 6) who was extremely pleased to be involved especially when the other Simon (Cooke) took his picture for posterity – and to try out his new camera! See p.40 for the names of the lucky winners.

Congratulations to those five. To the rest of you we'd just like to say one thing: Remember that questionnaires are somewhat like relatives at Christmas – no sooner do you get rid of one lot than along comes another taking up your time and energy when all you really want to do is slob out with a can and watch Coronation Street. So look out for another one soon.

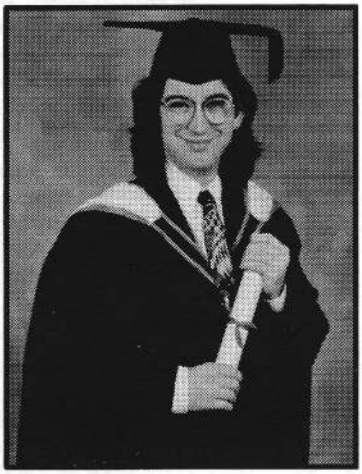
The more observant amongst you should by now have noticed the little attachment on this issue – this is all because of James Curry, who on his questionnaire under the heading "What would you like to see in future articles" put "Free chocolate bars in every issue". Unluckily for James, and anyone else with a sweet tooth, this is strictly a one off. (And for anyone who thinks we'll be daft enough to fall for the answer "Free £5 with every issue" – ***forget it pal!***)

Can we just point out here that we have in fact moved now, so if you want to get in touch with us, please make sure that you use the Squires Lane address (see the Contents page or the Subscriptions section for the full details). However, our mail is being forwarded for the next few months so if you've written to us at the old Dovey Close address don't despair because it'll still get here.

We'd also like to remind you that we welcome all kinds of input from our readers; not just articles, but also items for inclusion in the Letters, Helpdesk, News, Contacts and Mart sections. We'd really like to hear if you're working on any new software or hardware, so that we can include it in our SAMizdat section (which is our "vapourware" section for forthcoming projects and products under development) so don't be shy, write in to us today and you too could see your name in print!

And finally, seeing that Simon Cooke came up with all manner of weird and wonderful excuses for not chipping in with the editorial this time round, we (Maria and Martin) reckon that gives us all the excuse we need to embarrass him. For anyone who doesn't know, Simon has spent the last three years at UMIST doing a Physics with Electronics degree, and after recently graduating is now entitled to put BSc (Hons) after his name. Congratulations Simon, and well done!

Martin Rookyard, Maria Rookyard, July 1996


Would you trust this man to tie his own shoelaces?

Apologies

You may have noticed that this issue is a few weeks late. I'd like everyone to know that this is all my fault (Simon Cooke, that is). Work, graduating, learning the new DTP layout software for the magazine and increasingly chaotic deadlines set by the news-stand magazines I write for have meant that I've not been able to get this issue together in time for the deadline. It ain't Martin and Maria's fault in any way, and I'd like to apologise to you all for holding up this issue of the mag. Barring fire, flood, torrential rain of fish and other forteana, it won't happen again, I promise. Cross my heart.

Another matter comes to mind; that of Colin Anderton's review of Issue 1 of B.O.A.I. in FRED disk magazine. He mentioned that it was April and the second issue of the mag wasn't in sight. This is because we had a lot of problems with the printing of the first issue, so we decided to delay the launch until the Gloucester SAM & Spectrum show, but couldn't change the text as the final version was already at the printers. We're back on schedule now (though a few weeks late due to my own incompetence), so everything should be tickety-boo.

Anyway, enjoy this second issue of the magazine – it's a corker. Oh, and don't be shy – write us some letters for inclusion in the magazine, and send in your Feature ideas!

Love Simon

The SAM's Premiere PD Library

SAM PD

18 Mill Lane, Glenburn Rd, Old Skelmersdale, Lancs, WN8 8RH. Tel: (01695) 731163

We have a large selection of Public Domain (and copyrighted) software in stock to suit all tastes. A PD disk catalogue is available – send £1.00, or a formatted disk and an SAE. Also includes some useful progs and a Tetris game! All cheques should be made payable to SAM PD

SAM COUPE ARCADE DEVELOPMENT SYSTEM (SCADS) • £2.50
SCADS is now Public Domain. The disk contains the Designer and Supervisor programs, and comes with eight playable demos and a read-me file. The SCADS manual can be obtained from Revelation Software and also from us for £12.95 (200 pages, A4).

AMSTRAD NC100 NOTEPAD TO SAM • £1.50
Convert Notepad files to SAM format and vice versa.The files created can be used on most SAM wordprocessors. Excellent step-by-step read-me files will take you though all you need to know to transfer files and how to make up the necessary conecting lead.

PAW CONVERTER • £1.50
The Professional Adventure Writing system can now be used with full SAMDOS compatibilty. The original PAW, version A17C, on disk or tape is needed. Contact us if interested in this disk, but don't have A17C.

DJ0HF AMATEUR RADIO SUITE (Shareware) • Unregistered £1.50 • Registered £5.00
Transmit & receive radio data such as RTTY,CW,SSTV and FAX. The program can control an external TNC. To use, you need a SAM running MASTERDOS with a minimum of 256KB, 1 disk drive and the SAMBUS to provide timing signals. No other external hardware is needed to decode the signal from your amateur radio receiver.

UTILITY COLLECTION • £1.50
Includes Data Manager - database. Dir-Util - very nice disk manipulation program with pull down menus, handy Diary program, a full sector editor, a stand-alone auto Unerase program, which will restore any erased program without asking you for the file type. SAM-to-BMP converts SAM screens to Windows format graphic files, BMP-to-SAM does the reverse. Text-to-note converts PC text files to Tasword format, note-to-text does the opposite. Fred-to-text will decompress FRED disk magazine text files to normal text.

007 DISK DOCTOR • £1.50
Originally sold for £9.95, the disk contains five programs. Disk Doctor, Sector Editor, Close up, Map-it and Exchange. A readme manual file is also included on the disk.

007 DISASSEMBLER & 007 REVEAL • £1.50
Three disassemblers for disassembling code at different memory addresses. Reveal is a program for displaying text in code, find high score, names and hidden messages. Also on this disk is an updated disassembler version by Mike Haine, that is also a monitor program.

SAMART • £1.50
An excellent little Art Program, which is very user friendly. SAMArt uses two screens, a work screen and

MasterDOS and the SAMCo mouse.

F9 SOFTWARE – NOT PUBLIC DOMAIN!

KAPSA EMULATOR • £8.50
A Speccy emulator with lots of extra features. Use it to Snapshot or convert files. K.E.'s ROM lets you use every key of SAM's keyboard and has a help screen of the Speccy's keyboard. Three built in monitors (Z80-list, Vast and Wlezley - good for viewing graphics) with an MC tracer program for hackers. You also get Ancient Fighters and Tetris III free of charge.

ZIPPER (ARCHIVING PROGRAMS) • £8.50
Screen compressor, MC Data compressor and Disc compressor - the excellent Archiving program - archives selected files on average to 50%. FAST decrunching occurs automatically on loading.

SAMSPRITE • £9.95
A pointer driven Sprite Designer program, for designing 8x8 or 16x16 sprites. Animation feature and printed manual.

EDI-PRO WORD PROCESSOR • £13.50
Full WYSIWYG system, proportional printing, up to 15 fonts in one text file, pull down menu system, integrated font editor, imports text from Tasword and Outwrite, search & replace, screen saver, macros, 32 maths & other symbols, auto save, and many other functions we've not got space to mention! Edit up to 6 text files at once! Prints to Epson and compatible printers.

SAM MOD PLAYER v2.2 • £5.00
The latest version of this excellent MOD player, compatible with SAM Soundchip, EDDAC, Blue

NEWS

April Gloucester Show

Not many new releases or surprises in the way of new hardware or software was shown off at the Gloucester show, which is a bit of a shame, but never mind. What *was* there included Allan Skillman showing off with Xcoupe running on his laptop PC, Colin Piggott with a sampler for Quazar (£30, apparently it's very small, neat and compact, and what's more, it works very well. At high quality, you can fit a 1 minute sample in the SAM's memory), and Nev Young was showing off his hard drive as usual. Simon Cooke's Termite software was used to connect two SAMs together using the built-in chat software, but he couldn't rewire the phone-socket in the kitchens without getting into trouble, so there was no way it could be used to phone up Dalmation BBS. Maybe next time, eh?

Way of the Exploding Fist

Even though this is not a games-based magazine, we thought we'd tell you about a new one coming out soon. Although we have no name as yet, apparently it's very reminiscent of a popular fighting game. It'd be nice to see how fast it runs, and how large the sprites are, but that's all the info we have.

Cyberledbury!

David Ledbury is hoping to set up an Internet site soon for *Persona's Blitz* magazine or *Z2* soon, with Steve Pick doing all the hard graft on the *Blitz* one. Also, he wanted to point out that a new magazine – *Z2* – which he is producing with Mat Beal is available now, and contains "a combination of the best bits from *Fish* and *ZAT*". Issue 2 is almost out now, so check out the addresses in the *Contacts!* section of the magazine (p.41) for an address to write to.

All Wired Up

As announced on the Internet in various places, including the SAM Users Mailing list (see issue 1 for details of how to subscribe to it), Tim Paveley's SAM Coupé ScrapBook has changed location. You can now find it at **http://www.mono.org/~unc/Coupe**. Other SAM users web pages which have popped up in recent times include Steve Pick's site, which can be found at **http://www.ace.mdx.ac.uk/hyperhomes/houses/steven/home.htm**. More site addresses will be printed in the next issue of B.O.A.I..

Exploding Heads

Steve Nutting is now offering a scanning service for SAM owners (and in fact anyone else who wants to pay for the service). For £1, plus £1.50 postage and packaging, he can take an image which is up to a maximum of A4 in size, and create colour or black and white SAM screens from it. Higher resolution is available for black and white images, but if you're scanning for use on your SAM it won't make a difference. He can create roughly 32 SAM screens from an A4 image, all of which piece together mosaic-like. If you're interested, send an SAE for more details to Steve Nutting at 7 Narrow Close, Histon, Cambridge, CB4 4XX, or phone him between 6pm and 9pm from Monday to Saturday on (01223) 235150.

A Moving Experience

Okay, so we don't want to harp on about it, but there's sure to be some people who haven't noticed the liberal sprinking of new-address warning which we've scattered through this issue. You can now find *Based On An Idea...* at **38 Squires Lane, Tyldesley, Manchester, M29 8JF**. The phone number remains, as ever, **(01942) 886084.**

Strange Shortages

As far as we can tell, people are a little reluctant to provide material for the magazine. What can we do to convince you? Isn't two free issues per article enough? How about free adverts then? We've come up with a scheme where you can trade your free issues for adverts in certain circumstances. Phone us for details. Other than that, we'd like lots of letters, and info on things you're working on for the *SAMizdat* column. Basically, we need *YOU!*

All Roads Lead To ... Leeds?

Yep, on the horizon (and coming closer all the time) is the first Northern SAM and Spectrum Show – NSSS for short. Allan Clarkson, who also wears the hat of editor of *Crashed* magazine, is the guy who has organised it all, and people are turning up in abundance. If you find the prospect of travelling to Gloucester a bit horrifying, then you might just want to take the plunge and try this out. You can find out all the information you need about how to get there on page 33, and if you've bought this copy of the mag before the show, you'll find a money-off voucher in the envelope with it. Other than that though, we'll be there (well, at least one of us anyway), and we should have a report of the show in the next issue of *Based On An Idea...* which will be available from the end of October. *(I promise – Cookie)*

EMC Problems

New EC electromagnetic compatibilty rulings (EMC) which became effective at the start of the year have managed to severely disrupt the hobbyist and small-business electronics industry. Anything which is now sold has to have passed stringent (and expensive) tests to ensure that it doesn't conflict with the EMC rules. This has caused a number of companies to stop trading, as they simply cannot afford to have the tests done. The only solution to this problem seems to be to get around the rules by selling hardware in kit-form, which is inconvient for those who are not skilled with a soldering iron. While SAM production is not affected by the ruling at the moment, from January 1997 the rules will affect the production of the machine. Whether or not this will be the final nail in the SAM's coffin remains to be seen.

WE WANT YOUR NEWS!

You can email us at:

entropy@jumper.mcc.ac.uk

or post it to us at:

Based On An Idea, 38 Squires Lane, Tyldesley, Manchester, M29 8JF

If you prefer, you can phone any news to us at:

(01942) 886084

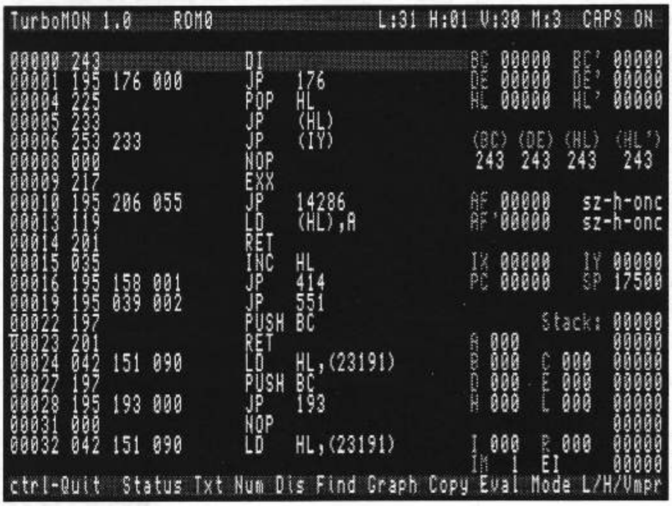
Reviews

What's the thing that every coder should have? A monitor of course (the debugging kind that is – though you might be lost without the other kind too)

Turbomon (and SC_Monitor – combined package)
Steve's Software, 7 Narrow Close, Histon, Cambridge, CB4 4XX. Tel: (01223) 235150 from 6pm-9pm, Mon-Sat.

Okay. Better get this out of the way first. This is a review of *Turbomon*. Now, this comes in a rather nice combo-package with Steve Nutting's SC_Monitor. The thing is, I'm only going to review Turbomon. No disrespect to Steve intended, but let's put it this way – I have a work disk on which you can find all manner of wierd and wonderful programming utilities. Turbomon is on this disk. SC_Monitor, however is not. I just can't get into using SC_Monitor for some reason – maybe it's because I've been spoilt by using Turbomon. Maybe not. Either way, this isn't getting me any nearer to reviewing Turbomon, but I just thought I'd let you know why the other program was … erm … absent from this review.

Turbomon has been written by Simon Owen, a little known coder (now in Entropy, on the strength of this software) who hasn't been seen to do much else on the SAM – he is, or rather was, quite active in the Spectrum scene in its hey-day though. It requires 48k of memory to sit itself in, with 16k for the code itself, and 32k for a special screen on which to put all



```
TurboMON 1.0    ROM0                        L:31 H:01 V:30 M:3  CAPS ON

00000 243              DI                     BC 00000   BC' 00000
00001 195 176 000      JP    176              DE 00000   DE' 00000
00004 225              POP   HL               HL 00000   HL' 00000
00005 233              JP    (HL)
00006 253 233          JP    (IY)             (BC) (DE) (HL)  (HL')
00008 000              NOP                     243  243  243   243
00009 217              EXX
00010 195 206 055      JP    14286            AF 00000     sz-h-onc
00013 119              LD    (HL),A           AF'00000     sz-h-onc
00014 201              RET
00015 035              INC   HL               IX 00000     IY 00000
00016 195 158 001      JP    414              PC 00000     SP 17500
00019 195 039 002      JP    551
00022 197              PUSH  BC                       Stack: 00000
00023 201              RET                    A 000          00000
00024 042 151 090      LD    HL,(23191)       B 000   C 000  00000
00027 197              PUSH  BC               D 000   E 000  00000
00028 195 193 000      JP    193              H 000   L 000  00000
00031 000              NOP                                   00000
00032 042 151 090      LD    HL,(23191)       I 000   R 000  00000
                                              IM  1   EI     00000
ctrl-Quit  Status Txt Num Dis Find Graph Copy Eval Mode L/H/Vmpr
```

Turbomon's main screen in all of its glory. On the right-hand side are the current register values, flags, PC and SP positions, the last eight words on the stack and any other salient information. On the left-hand side you can see a disassembly of the ROM. It isn't just a disassembly though – when this screenshot was taken, Turbomon was in single-step debugger mode. At the top of the screen is the current paging status of the monitor, and at the bottom, a quick and helpful reminder of what it can actually do. Ah, bliss. And it's damn good at what it does, too!

of the status info. This means that unless you're writing something like Lemmings (in which case you'll probably have knocked up some kind of development system using another machine as well) you'll always have room to sit this program in because it's so compact.

The most powerful feature of Turbomon is its built-in single-step debugger. It's incredibly fast (Simon has optimised everything in this program – I've seen the source code, and it's a nightmare of self-modifying code and other tricks), and you don't even have to step through the code instruction-by-instruction – you can set up conditions which will stop the debugger, and then set it running, which is absolutely excellent for debugging – it really helps make diagnosing bugs much easier.

There are also text and numerical data editing modes (great for hacking), a plain-vanilla disassembler (but an incredibly *fast* one), block memory fill and copy commands, memory search functions, a graphical memory usage display, a built-in calculator… there's pretty much everything in here. In fact, the only way I can fault it is that it can't handle external memory, and it doesn't handle the refresh register properly – apparently this would cause too much of an overhead, so it's updated from the *real* R register, which effectively gives a random number every time that register is read from. Understandable, and this should be fixed by giving the option to update the R register correctly, at the expense of a time penalty in emulation.

Turbomon comes with a rather nifty little manual. It's so well presented that I can't readily believe the author's claim that it was put together on Write for Windows, but there you go. The really groovy thing about the manual is that in the back of it there is a large list of opcodes. In fact, it's a list of *every single possible* opcode you can come up with – including *every* undocumented one. It's great fun if you're looking for something to confuse people who, say, don't have Turbomon and so won't be able to decode these opcodes properly using their disassemblers. However, with the accelerator still looming on the horizon, it might be best to steer clear of these – namely because most of them won't work with the Z380 processor.

In conclusion, I've got my copy. If anyone wants to try and take it off me, then they're going to have a battle on their hands. It's great for hacking other people's code, as well as for getting really stubborn and hard-to-find bugs out of your own. Well polished, easy to use, fast and efficient, Turbomon is the only monitor program you should have on your work disk.

RATING:
5/5

SUMMARY:
Take a lot of effort out of debugging your code and buy this immediately. Not only that, but it's great for hacking too!

Simon Cooke

Subscriptions

THREE reasons why you should subscribe to *Based On An Idea...*

It's Cheaper!

To buy each issue separately costs £2.00 x 4 = *£8.00*
To buy a year's subscription (four issues) costs only *£6.00* - an amazing saving of *£2.00!*

Less Anxiety!

We send out your copy of the mag as soon as it comes back from the printers – no more wondering "Is the latest issue out yet?" – you'll know because it will pop through your letter box as soon as we can possibly get it to you!

Guaranteed: No Furry Animals

No matter what depths we end up stooping to, we can guarantee that when you open your copy of *Based On An Idea...* **you will never ever find a small hamster called Gerald quietly chewing its way through the pages. Unless someone slips him in at the sorting office that is. In the unlikely event of this happening, we recommend that you attempt to wean him away with some fresh lettuce.**

To subscribe to *Based On An Idea...* **all you need to do is send your name and address with a cheque for £6.00 made payable to M. Rookyard to the address below. Subscription wrangler extraordinaire Maria Rookyard will take your details and ensure that you receive your very own pristine copy of the very latest issue of** *Based On An Idea...* **with a minimum of fuss and effort on your part. Please state which issue you want your subscription to start with in your covering letter.**

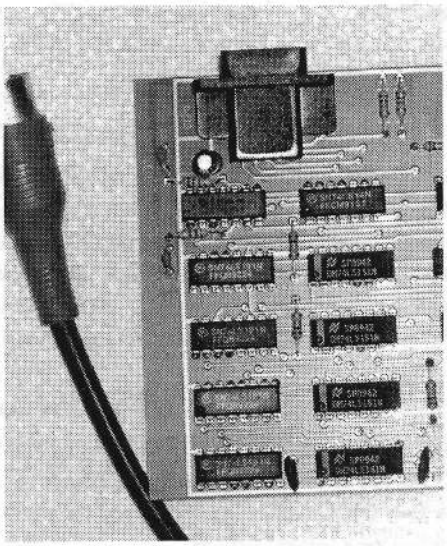
Based On An Idea...
38 Squires Lane, Tyldesley,
Manchester, M29 8JF

Of Mice And Men...

Part 2: The SAM Mouse Hardware Explained

In part one of this article, we explored how to read the SAM mouse using software routines. Part two explains how the box of tricks connected between the mouse and the SAM actually works...

I've got some bad news for you. Remember how mind-numbing last month's SAM Mouse article was? Well, this one is even more mind-numbing than that (though if you're technically minded you'll probably lap this up). For a start, it's the electronics side of the interface, and as such it's something that most programmers never concern themselves with. Electronics enthusiasts should find this quite interesting, however, as there are a number of nifty tricks which Bruce has employed to get an Atari ST mouse to work with the SAM.



Ancient History

When the SAM was on the drawing board, the idea was that MGT would create its own custom ASIC to drive the mouse. Unfortunately, before they had time to do that MGT folded, and the SAM Mouse never appeared – at least, not in the intended form. Because of growing demand for some form of mouse for the SAM, SAMCo decided to take the plunge and design an interface which would allow an off-the-shelf mouse to work with the SAM. Unfortunately, this lead to a rather large box being bundled in with the software to run it.

Meeces To Pieces

Your average mouse works in a very imaginative way. There are a couple of wheels which are turned by the big rubber ball in the base of the mouse; one for vertical motion, and one for horizontal motion. There is also a sprung roller which keeps the ball in contact with the rollers connected to the wheels.

At the end of the roller, a disk with teeth cut out of it can be found. At the base of the wheel (usually in two different locations) there are a couple of sensors – infra-red light-emitting diodes, with photo-sensitive cells directly opposite them, shining through

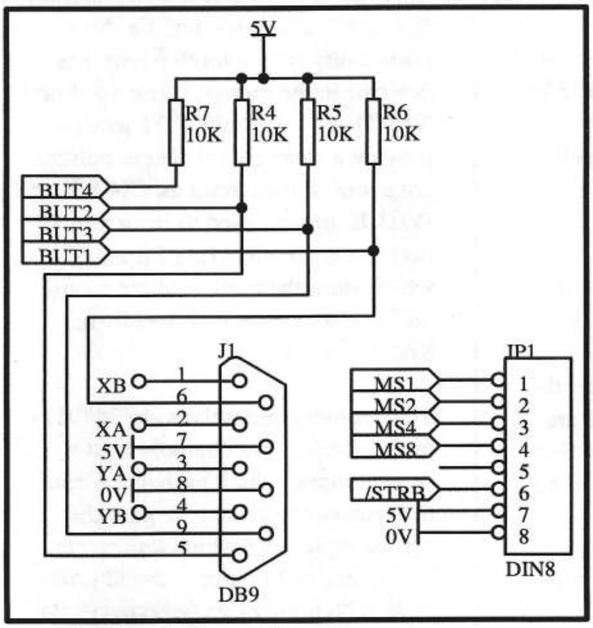


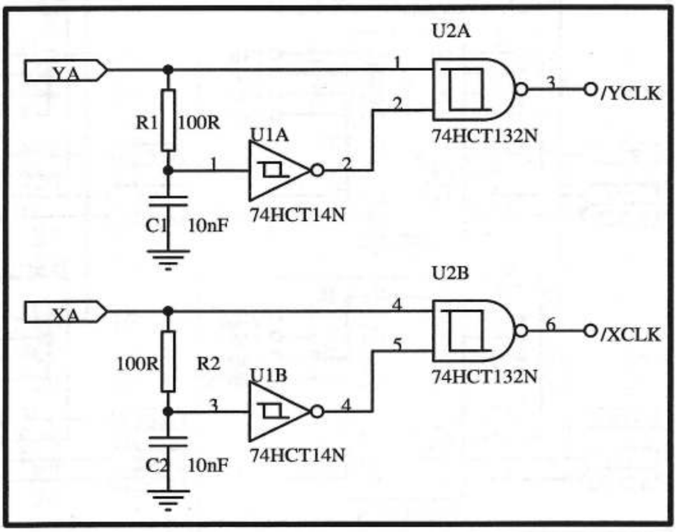
Fig 1: Input / Output Connectors

the holes in the teeth. Because of the geometry of the teeth and position of the sensors, it is possible to tell which direction the mouse is moving in at any time, and by how far. The sensors themselves generate a train of pulses; a main signal is generated (**XA** and **YA**) which indicates motion, and a quadrature signal (**XB** and **YB**) which is generated 90° out of phase with the main signal. This quadrature signal indicates which direction the mouse is moving by whether or not it leads or lags the main signal. Figure 1 shows where the signals from the mouse come into the interface box, and where they come out of it.

The **5V** and **0V** lines going into the mouse (on the 9 pin D connector) are what powers the device. The other lines (**BUT1, BUT2** and **BUT3**) are connected to a series of microswitches inside the mouse itself – usually there are only two of these, less commonly there are three. The interface aappears to cater for up to four buttons, but the fourth button line is tied to **5V** using a 10K resistor to ensure that a logic 1 is received on that line when it is read.

The other connector (8 pin DIN) is the one that plugs into the SAM. There are four output lines (**MS1, MS2, MS4** and **MS8**) which connect to the keyboard lines, and it is these which are read when you read the **RDMSEL** port. The **/STRB** line is pulled low

Fig 2: Mouse motion detection



when this port is read, and is used to signal to the interface that mouse data is required. On this connector, the **5V** and **0V** lines are used to power the mouse interface and the mouse itself from the SAM.

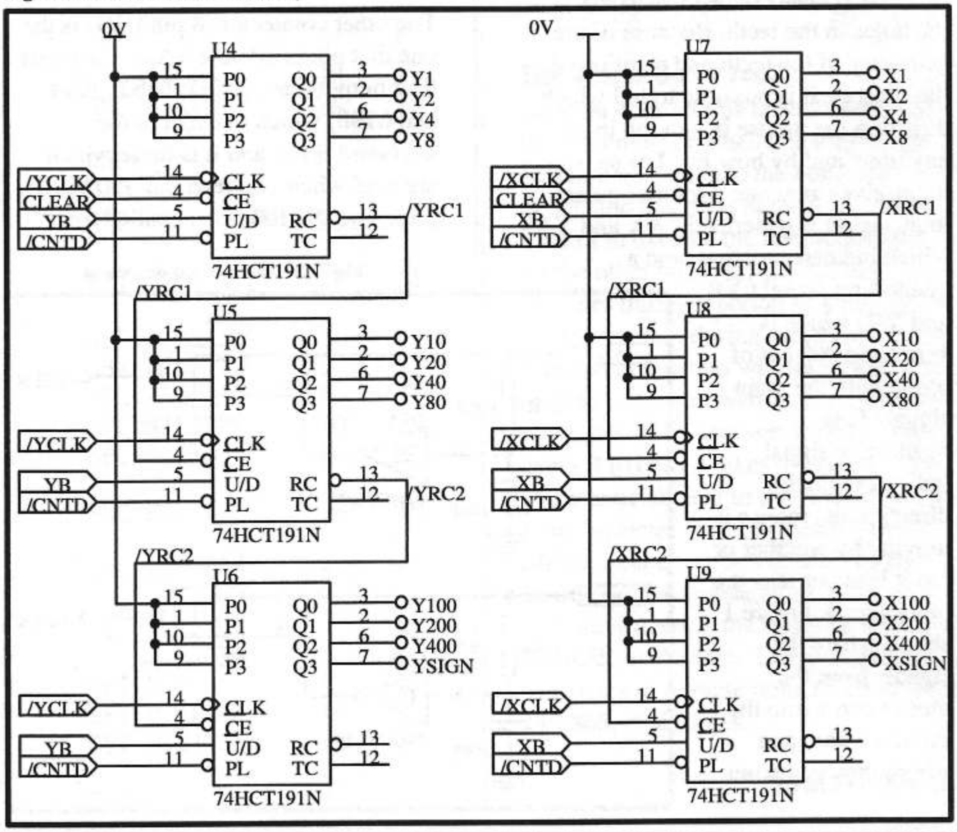
Clocking On

Figure 2 shows the quadrature decoding circuitry. **XB** and **YB** are taken through directly to the counter circuitry to determine which direction the mouse is being moved in – if they are low, the mouse is moving in a positive direction, if high it is being moved in a negative direction.

The circuit takes the main signal lines **XA** and **YA**, and produces a short pulse every time a tooth passes the detector in the mouse, using a Schmitt NAND and a Schmitt NOT gate to provide a clean signal. These pulses come out of the circuit as **/XCLK** and **/YCLK** and are used to increment or decrement counters (see Figure 3) which store the motion of the mouse until the next time it is read by the SAM.

The counters themselves are 74191s – bidirectional 4-bit counters with a ripple output, which basically means that you can use one to trigger the update of another which represents higher digits. In order to avoid problems with logic races (where signals

Fig 3: Motion Counter Circuitry



arrive at different times, confusing the outputs) which might occur if you used, say, the most significant bit of the output of a counter to clock a higher counter, the ripple output allows you to use the same basic clock signal for all of the counters. The ripple output is then used to enable higher counters to be updated when a lower counter rolls over to zero. To illustrate this a little, if one counter reached 15, the ripple output would go low, enabling the higher counters. When the next clock pulse arrives, the counter wraps around to zero again, clearing the ripple counter output. However, before the ripple output goes high again, the higher counter is clocked and increments by one.

The ripple output is marked as **RC** on the diagram. As I have said before, the **XB** and **YB** signals control the direction the counters count in. So this leaves only two signals which are unaccounted for – one of which is the **PL** line. This is used to program the counters with a specific starting value – in this case zero, as lines **P0**, **P1**, **P2** and **P3** on each of the counters are tied to ground. The line which goes into the **PL** input comes from the SAM side of the interface logic – ie the bit which lets you read the data from the counters. This is the **/CNTD** line, and we'll cover that in a moment.

The other signal which we've not looked at yet is the **CE** input on the least-significant X and Y counters. Normally this is where the ripple input comes in, but on the least-significant counters, there is no lower-value counter to take the ripple output *from*. The signal which comes in on this line is a **CLEAR** signal, which (as with the **PL** input) is part of the SAM interface logic. The **CLEAR** signal is used to stop the counters from being updated while the SAM is reading them. Any information about the motion of the mouse will be lost during this time, but that doesn't matter, because the amount of time it takes to read the mouse is a matter of microseconds.

Control Freak

The SAM interface logic is the niftiest piece of design on the board, and is a paradigm of cost-cutting and lateral thinking. Figure 4 shows this circuit, which is the heart of this part of the interface.

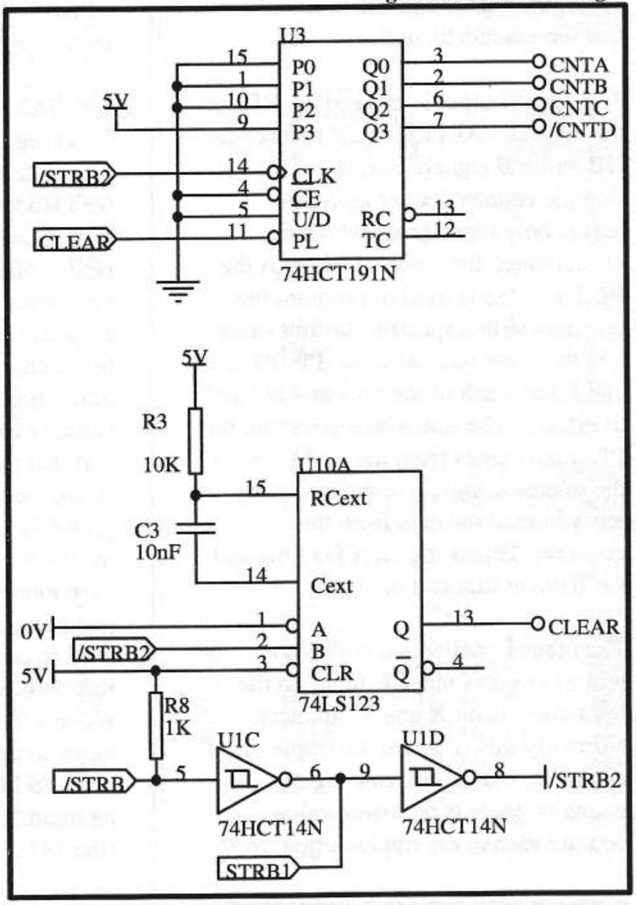
The SAM reads the interface by checking the **RDMSEL** port. When you do this, it generates a strobe signal (**/STRB**), which although normally high, goes low for the duration of the read. This signal is taken into the circuit via a 1K pull-up resistor (which ensures that the signal is either high or low, and not in-between), into a Schmitt inverter. This chip doesn't actually have to be a Schmitt one – it's just that there's about six per chip, so it would be a shame to waste some perfectly good gates which we have available when we can use them and save money in the process. This generates a **STRB1** signal, which is used to control the output stage of the interface, which we'll cover in a second. This signal is then inverted again to provide a slightly delayed and clean **/STRB2** signal, which is used as an input to a one-shot monostable chip (the 74123) and to a counter which controls exactly what is sent to the SAM at what time during a read.

Although I have been told that using a monostable in this way is slightly naughty, it's a nice and cheap – if not noise immune – way around the problem of controlling the output logic. As an aside, it is this monostable which causes the infamous "mouse walk" problem, where a mouse appears to be unresponsive. The solution? Redesign the board, or put some *very* big capacitors into your SAM motherboard across the power supply, as the standard SAM one tends to be very noisy and prone to picking up all kinds of spikes and glitches.

The one-shot monostable chip provides the **CLEAR** signal to the rest of the system. Normally (i.e. when inactive) the **CLEAR** signal is high. When the strobe signal arrives from the SAM, the one-shot is triggered, and its output goes low for a *minimum* of ~30µs. I say minimum because any reads from the SAM while the output of the monostable is still low cause the one-shot to remain low for another 30µs from that moment. Continuous reading of the interface will causes this signal to remain low indefinitely. The width of this pulse is determined by the 10K resistor and 10nF capacitor network which feeds into the **RCext** and **Cext** inputs of the monostable. (In fact, if we really wanted to pin-down a culprit for the noise in the system, it's this RC network… though the monostable itself is traditionally a very noisy chip anyway).

The **CLEAR** signal indicates whether or not a read from the SAM is currently in progress. As it's a minimum of ~30µs long, this means that if you take longer than 30µs between reads of the interface, you'll have to start again from scratch, as the monostable will have reset, as will the counter which

Fig 4: SAM Interface Logic



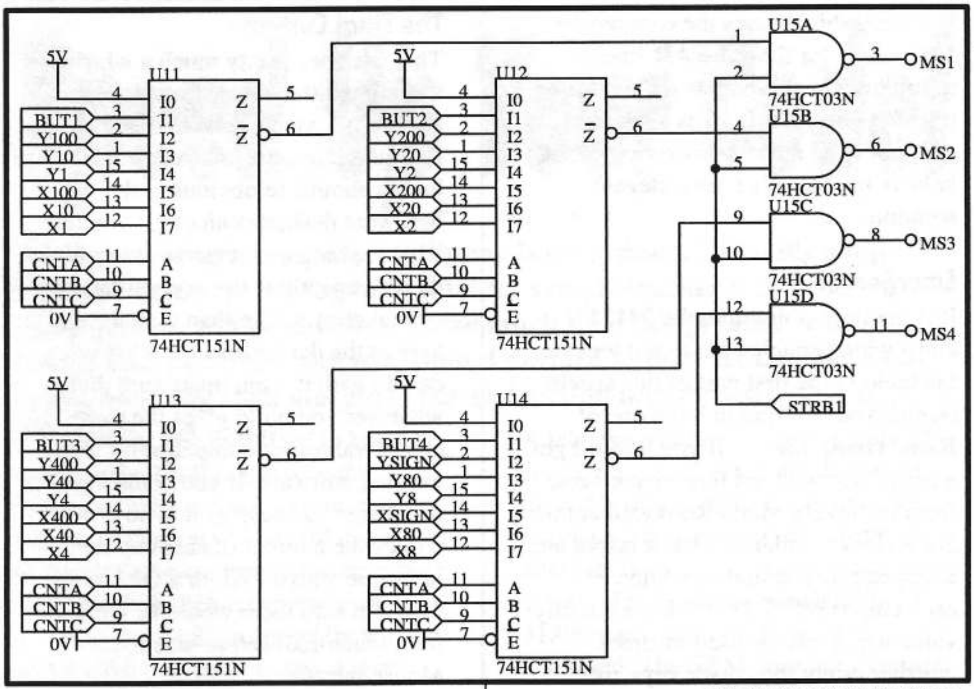


Fig 5: Output Logic Stage

handles the output logic.

Speaking of which, it's about time we covered the counter. The **CLEAR** input is used to reset the counter to **%1000** (in binary), which gives the initial **CNTA**, **CNTB** and **CNTC** outputs as zero, and **/CNTD** as 1. (The inputs are all tied to **0V** except for **P3** which is tied to **5V**). The **U/D** input is tied to **0V**, which means that the counter is continuously counting upwards, and so counts from (in decimal) 8 to 15, then round to zero again. The **CE** input is also tied to zero, which means that whenever a clock signal is received (on the **/STRB2** line) the counter will increment. The **CLEAR** input is low while the SAM is reading the interface, and the **PL** input requires a high level to program the counter, so this means that the counter is free to count upwards while the SAM is reading, and 30µs after it finishes it is reset back to the inital value of **%1000**. After each read, the **/STRB2** line, which was low during the read, goes high again, clocking the counter and incrementing it by one to the next value. Phew.

Basically what this particular counter does is to provide a signal to a series of 74151 chips, which are 8 to 1 line multiplexers – that is, the output of the counter determines which one of 8 different inputs are sent to the output of the multiplexer (see Figure 5). What is interesting is that the **/CNTD** output of the counter is used to reset the motion counters to zero. When a read is in progress, **/CNTD** is always high (this is why the counter is initially programmed to **%1000**), until the very last read which clears the counters to zero, ready for the **CLEAR** line to return high and re-enable the motion-tracking counters. It takes a while to get your head around this concept, but believe me that it's a very elegant solution.

Emergence!

It is the lines going into the 74151 chips which enabled me to put together the table in the first part of this article (which you can find in issue one of *Based On An Idea...* If you haven't got a copy, then why not buy a back-issue from the lovely Maria Rookyard at the *Subscriptions* address? Back issues are £2.00 each). The first input line of each chip is set to **5V**, giving a dummy value which can be used to check whether or not the cursor keys are depressed, which would screw-up the incoming mouse data. The next in the sequence is the button status, then the most-significant Y nibble, next significant, then least significant, and again for the X values. To ensure that these values (which are generated all the time) don't screw up with the operation of the keyboard, the actual output of the values onto the keyboard lines is controlled by four NAND gates. These are combined with the **STRB1** line which we generated from the **RDMSEL** strobe line in Figure 4, such that if a read is not currently in progress, the output of these gates will always be **1** or *high*. This is done because high values do not disturb the keyboard matrix – depressed keys are represented by *low* values – and so with this our exploration of the interface is complete.

The Final Curtain

This has been pretty much a whirlwind trip around the mouse interface, and hopefully I've not missed anything out. By including schematic diagrams, with luck it should be obvious to the hardware designers amongst you just what is going on – even if my prose doesn't exactly make everything crystal clear. Other than that, have a look at the design and see what you can do with it – but make sure that whatever you build gives the same output values for compatibility with existing software. If you could keep the design as cheap as it is now, and reduce the number of chips by replacing some with a PAL or a GAL chip then I'm sure there would be a market for a much-reduced-in-size SAM Mouse interface.

In issue 4 of *B.O.A.I.*, we'll be printing details on how to write a mouse driver for PC serial mouses *(sic)* if you've got a comms interface. Until then, this concludes everything you need to know about the SAM mouse.

Simon Cooke

ERRATA

On p.24 of issue 1 of *Based On An Idea...* I printed that it was possible to optimise the mouse reading routine by removing the final LD (HL),A from it. It seems that I had already performed that particular optimisation to the routine. Therefore, you should ignore that comment, and use the routine as it stands. Sorry if this confused you.

How To Read *

No, no, no – this isn't a course teaching you how to read. A written course would be rather pointless in that case wouldn't it? What I am going to do is try to explain the structure of an MS-DOS disc (referred to as DOS from now on).

Although there are many people with a hatred of Microsoft out there, I'm going to stick my neck out and say that I like the way DOS discs are structured.

The DOS format has the following advantages:

- it has a boot record containing information on the structure of the disc and therefore allowing variable disc sizes (even hard discs!);
- a subdirectory structure which lets you have a (theoretically) infinite number of files on a disc (you are not limited by the amount of tracks that you allocate to the directory when formatting);
- better use of sector space by having a FAT (file allocation table) showing the DOS where to find the sectors belonging to a certain file. This way the full 512 bytes of a sector can be used instead of the 510 used by SAMDOS. However, for this flexibility there is a price to pay – see disadvantages;
- the disc is used cylindrically instead of as two separate sides. This means that instead of moving to the next track after reading a track, the other side of the disc is read. This leads to less stepping and should therefore mean a faster disc read. The funny thing is that my SAM reads a PC disc faster than my PC does!

Unfortunately there are also a number of disadvantages inherent in the MS-DOS structure:

- double density discs are only formatted to 720K whereas SAMDOS discs are formatted to 780K. Historically it was considered safer to only use 9 sectors per track as opposed to SAMDOS's 10, and as such MS-DOS is limited to 9 sectors per track. Theoretically discs can be formatted in an MS-DOS structure with 10 sectors per track, giving more than the 780K available on a SAM disc due to a less wasteful directory structure, but MS-DOS systems would not be able to read them due to sloppy programming on Microsoft's part;
- to prevent the FAT growing too large, the FAT allocates space in terms of clusters instead of sectors. A cluster consists of a number of adjacent sectors (the exact number per cluster depending on the size of the disc). A 720k formatted disc has two sectors per cluster, and therefore the space

***MS-DOS Discs**

saved by using a FAT is therefore reduced if you have files which only use a few bytes of the last cluster.

Now for a quick recap of all the things to do with the disc format. If we look at the way a disc is formatted, a disc has two sides, 80 tracks per side and typically 10 sectors per track. When referring to a specific physical sector I will do this as follows: (side, track, sector). Side can be either 0 or 1 (but could be more for a hard disc), Track is in the range 0 to 79 (even though most SAM floppy drives can step to track 83) and Sector is in the range 1 to 10 (even though you could format a 3.5" disc with a lot more smaller sectors, or with five 1K sectors and one 512 byte sector to give 960K of space per disc!).

First up is the typical SAM disc. A disc consists of two sides, each side is split up into 80 tracks, each track is split up into 10 sectors. The first 4 tracks on side 0 are used to store the directory, and the rest is used to store the actual data part of the files.

Secondly, we have the standard 720K DOS disc. A DOS disc consists of 80 cylinders (a cylinder is a track, but takes into account all possible sides of the disc). Each cylinder consists of 18 sectors, with 9 on each side. DOS also uses the term logical sector. A logical sector is a number from 0 to the total number of sectors on the disc. This makes it easier to refer to data on the disc without needing to know the sector, track and side where it lies. Logical sector 0 is the first physical sector, the next one is the next sector on that track, and when all sectors on that track have been used then the next logical sector is on the other side of the disc. For example:

logical sector	physical sector
0	0,0,1
8	0,0,9
9	1,0,1
17	1,0,9
18	0,1,1

To convert a logical sector to a physical sector you can use the following formula:

physical_sector = logical_sector MOD sectors per track + 1

physical_track = logical_sector DIV sectors_per_track DIV number_of_sides

physical_side = logical_sector DIV sectors_per_track MOD number_of_sides

In machine code this could look like this:

```
;on entry DE = logical sector [0..65535]
;on exit  D =  physical track [0..79] + 128
;              for side 1
;         E =  sector [1..max sector]

         EX DE,HL
         LD BC,(sectors.per.track)
         ;bytes 24+25 of boot sector
         LD DE,0
         XOR A
rlsdiv:  SBC HL,BC      ;a more
                        ;elegant divide
                        ;routine could be
         INC D          ;used
         JR NC,rlsdiv
         ADD HL,BC
         DEC D
         LD E,L
         LD A,(sides)   ;byte 26 of boot
```

```
                    ;sector
        CP 2
        JR NZ,$+4
        RRC D
```

The first logical sector (= first physical sector – side 0, track 0, sector 1) is called the boot record, and this contains information about the physical size of the disc (number of sides, number of tracks, number of sectors per track, sector size, cluster size), where the directory can be found and also where the FAT can be found. The values usually stored for a given disc size are shown in Table 1.

A DOS disc can be split up into four sections (as the above table shows):

- the boot record
- the file allocation table (+ back up)
- the root directory
- the data area

To keep some kind of structure in this article I will explain the four disc sections in the above order.

Boot Record

The boot record contains all the information on the format of the disc in the drive. Apart from this useful information it also contains some useless (for us anyway) startup code allowing either a PC to boot off this disc or give a message telling you that this ain't no system disc. Table 2, on the next page, contains all the useful bits that can be found in the boot record.

Using the boot record you can navigate around the four main sections of a disc. The easiest bit is the boot record, this is always located on logical sector 0. This explains why DOS discs cannot be formatted if track 0 is damaged. The next section, the first copy of the FAT can be found by adding the number of reserved sectors (bytes 14,15) to logical sector 0. To get to the root directory (which can contain a set maximum number of files) you add the sectors per FAT (bytes 22,23) times the number of FATs (byte 16) to the starting logical sector of the FAT. To get from the root directory to the data area requires a slightly more complex calculation, add 32 (bytes per directory entry) times the number of root directory entries (bytes 17,18) divided by the number of bytes per sector (bytes 11,12).

Here are all the useful little formulas, where offset is the address at which the boot record is loaded:

```
boot_record = 0
FAT = boot_record + DPEEK (offset + 14)
root = FAT + DPEEK (offset + 22) * PEEK
(offset + 16)
```

TYPICAL DISC VALUES

Disc	Disc	Sectors	Media	Cluster	Sector location ranges				
Size	Type	per Trk	Descrpt	Size	Boot	FAT1	FAT2	Root	Data Area
360 kb	5.25" DD	9	FD	1024	0	1-2	3-4	5-11	12-719
720 kb	3.5" DD	9	F9	1024	0	1-3	4-6	7-13	14-1439
1.2 Mb	5.25" HD	15	F9	512	0	1-7	8-14	15-28	29-2399
1.44 Mb	3.5" HD	18	F0	512	0	1-9	10-18	19-32	33-2879
10 Mb	hard disc	17	F8	4096	0	1-8	9-16	17-48	49-20720
20 Mb	hard disc	17	F8	2048	0	1-40	41-80	81-112	113-40457
60 Mb	hard disc	17	F8	2048	0	1-118	119-236	237-268	269-120596

Table 1: Typical disc values

BOOT RECORD INFO			
Name	**Bytes**	**Type**	**Description**
Unknown	0-2		This space contains a jump instruction to a portion of code later in the boot-sector, which loads the rest of the DOS
System ID	3-10	text	ID supplied by format program - no function, describes which version of DOS the disc was formatted with
Bytes per sector	11,12	word	Usually 512 bytes. Can also be 128, 256 or 1024 bytes when using a floppy.
Sectors per cluster	13	byte	Usually 2 on a floppy disc. For hard drives it is desirable to increase this to prevent the FAT from exploding
# of reserved sectors	14,15	word	How many sectors are reserved for boot record, usually 1
# of FATs	16	byte	How many copies of the FAT are there. Usually there is at least one extra copy since the disc structure is lost if the FAT is damaged somehow. Although MS-DOS updates these copies, it cannot use them to recover a disc. (Sloppy programming?)
# of root dir. Entries	17,18	word	How many entries are present in the root directory. Each directory entry requires 32 bytes so multiplying this with 32 and dividing by the sector size will give the amount of sectors used by the root directory.
Total sectors on disc	19,20	word	Although not strictly necessary you can use this number to check if you are dealing with a DOS disc.
Format ID	21	byte	This is the media descriptor (see Table 1). You can use this to check what kind of FAT is used (12 bits or 16 bits). The media descriptor is also stored in the first byte of both FATs.
Sectors per FAT	22,23	word	Number of sectors used by each FAT, the data area of the disc therefore starts at logical sector: [14,15] + [16] * [22,23] + 32 * [17,18] / [11,12]
# sectors per track	24,25	word	Needed to convert logical sectors to physical sectors
# of sides	26,27	word	Also needed for logical to physical conversion
			The rest of the data is extended information for certain versions of MS-DOS, plus executable code to load the operating system
MS-DOS ID tag	510,511	word	Tag indicating that this disc is an MS-DOS disc. Can be used to decide whether or not the disk is MS-DOS or SAMDOS format, as this value can never occur on a SAMDOS or MasterDOS formatted disc. For MS-DOS discs this is always &AA55

Table 2: Boot Sector data structure

```
data = root + 32 * DPEEK (offset + 17) /
DPEEK (offset + 11)
```

The following conditions can be used to check if a disc is a DOS disc:

```
dpeek (offset + 26) > 0
        (at least one side needed)
dpeek (offset + 19) > 0
        (at least one sector on disc)
dpeek (offset + 24) > 0
        (at least one sector per track)
dpeek (offset + 19) MOD dpeek (offset +
24) = 0
peek (offset + 21) = peek (FAT + 0) = peek
(FAT + dpeek (offset + 22))
```

[*Editors note:* A more reliable method to check for an MS-DOS format disc (although it will fail on an Atari ST one) is to check for the MS-DOS ID tag, the information for which I have added to the bottom of the table on the previous page.]

That covers the boot record. Now on to the file allocation table.

File Allocation Table

This can be seen as a road map telling MS-DOS where to find its files. The FAT contains as many entries as there are clusters. DOS uses the FAT to find out what cluster it needs next. The directory entry contains the first cluster. Once this cluster has been read, DOS uses the cluster number to index the FAT, the content of that address is then the next cluster. This way you end up with a chain of clusters that together form a file. Certain values for a FAT entry have a special meaning – see Table 3.

There is one minor snag with Table 3. It assumes that FAT entries are 16 bits (= 2 bytes). This was thought to be too wasteful for floppy discs, so on floppies the FAT entries are 12 bits (= 1.5 bytes). 12 bits allows us 4096 FAT entries which is ample for a floppy disc. For a 12 bit FAT system the above table is still valid, simply knock off the leading digit of all the values, a bad cluster is now referred to by FF7 (hex) instead of FFF7 (hex).

You may now wonder how you can tell if a FAT is 12 bits or 16 bits. The general rule is that floppies use 12 bits and hard drives use 16 bits. You can use the following formulas to grab a FAT entry, offset is in both cases the address at which the FAT is loaded.

12-bit FAT:

```
LET x = 1.5 * cluster
```

Description of FAT entries

Value (hex)	Meaning
0000	cluster available
0002-FFEF	cluster in use (points to next cluster in file)
FFF0-FFF6	reserved
FFF7	bad cluster, do not use
FFF8-FFFF	last cluster in file

Table 3: FAT entry meanings

```
IF x = INT x THEN
        LET cluster = dpeek (offset + x)
        BAND &0FFF
ELSE
        LET cluster = DPEEK ( offset + x -
0.5 ) DIV 16
```

16-bit FAT:

```
LET cluster = DPEEK (offset + 2*cluster)
```

Root Directory

The root directory can contain a limited number of file entries. The file entry gives information on the name of the file, when it was created, what size it is and also the first cluster. By using the first cluster to index the FAT you can then get the next cluster, and so on. The directory entry information is stored as detailed in Table 4.

DOS doesn't use file types in the same way that SAMDOS does. The filename extension (the three characters after the ".") is used to give the user some idea of what the file can be used for. The extension only gives an indication of what the file is – files often aren't what they claim to be.

Common extensions are:

- .BAS: basic program;
- .TXT/.DOC: text file;
- .GIF/.PCX/.BMP/.JPG: screen images;
- .MOD/.WAV/.MID: music/sound files.

The attributes byte gives some additional information on the properties of the file entry. Each bit represents a property. The most important bits for us are bits 3 and 4. If bit 3 is set this indicates that this entry is the volume label. If bit 4 is set this indicates that this entry is a subdirectory. A subdirectory contains data structured in the same way as in the root, each entry consisting of 32 bytes. A subdirectory always consists of two compulsory entries which are created when the subdirectory is created. They are the current directory (single dot) and the parent directory (double dot). A subdirectory is stored just like a regular file, when it is first created one cluster is allocated to it. As soon as one cluster is filled with file entries, additional clusters are allocated to the subdirectory entry by finding the next free cluster and adding this to the cluster-chain in the FAT. By allocating clusters to the subdirectory when necessary it is possible to have a virtually unlimited number of files on a disc. Bits 0,1,2 and 5 of the attributes are set if a file is read-only, hidden, system or archived respectively. A normal file only has bit 5 set. Bits 6 and 7 should always be 0.

Since PC's were fitted with clocks (or

DIRECTORY ENTRIES

Description	Bytes	Type
Filename	0-7	text
Extension	8-10	text
Attributes	11	bit
Reserved	12-21	possibly used by OS/2
Time	22-23	encoded
Date	24-25	encoded
Starting FAT entry	26-27	word

Table 4: Directory Entry Format

at least the facility to add them) right from the beginning, date stamping was incorporated into MS-DOS as well. Date stamping means that the date and time that the file was created are saved along with the file in the file entry. The time is stored in byte 22 and 23, the date in byte 24 and 25. For ease of explanation I will refer to the bits in these byte pairs as bits 0 through 15, with the lower 8 bits belonging to the lower byte. Minutes are stored in bits 5 through 10, hours are stored in bits 11 through 15 in 24-hour notation. The day of the month is stored in bits 0 through 4, the month is stored in bits 5 through 8, and the year is stored in bits 9 through 15. The year is stored as an offset from 1980 (surprise, the year the first PC came out). So the year 1996 is represented as 16 (1996-1980=16).

On a somewhat unrelated matter to this article, if you have Masterdos and would like to date stamp your files, but have not got a SAMbus then you could use the following routine which I put in my AUTO file. To view the dates use DIR DATE.

```
IF PEEK DVAR 150<>0 THEN
        INPUT "date (ddmmyy): ";LINE
d$
        IF d$<>"" THEN
                POKE DVAR 150,0
                DATE d$
```

The starting FAT entry points to the first cluster of the file, which is also the index for the FAT. At that point the FAT will then tell you the next cluster that belongs to the file (or subdirectory!).

The file size tells you how large the file is. The nasty thing about the file size is that it is a 32-bit number - not the nicest of things to work with on our nice little 8-bit machine. If you want to be lazy and don't want to bother with big numbers, *and* you only want to read files then you can just keep reading the file until you get an end of file marker when getting the next cluster from the FAT.

The End?

Well, that should tell you enough to explore those DOS discs. Until a DOS comes out for the SAM which recognizes both formats you will probably have to make do with KE_DISK. If you are writing a utility which makes use of files often found on the PC it might be an idea to incorporate automatic DOS detection allowing files to be loaded directly off a PC disc (as the SAM MOD player does! – *plug! plug!*). So all I have to ask now is who's going to write a really decent DOS which will also support the SAM hard drive interface?

Stefan Drissen

Editor's Addendum

Because we've not got a *SAMizdat* section up and running yet, this seems to be the most appropriate place to put this. *Termite* (Simon Cooke's Comms package) should come with a built in BIOS which will allow people to access MS-DOS, SAM and PRODOS format discs transparently (well, to an extent – file name conventions might get in the way a bit). This BIOS will then form the core of a new DOS (provisionally called E-DOS). When will it appear? Who knows… More info next issue.

QUAZAR

Colin Piggot • 204 Lamond Drive • St Andrews • Fife • KY16 8RR

16 bit sound strikes the SAM

The **Quazar Surround** is a powerful sound card for the SAM range of computers. It offers 6 sound channels, at a quality of up to **16 bits – matching CDs and top PC soundcards.** (The SAM sound chip can only play digital sound at 4 bit quality). Also, the **Quazar Surround** can play in full **surround sound.**

The **Quazar Surround** has two modes of operation, which can be selected from inside programs by means of a software switch. Mode 1 offers 6 channels of 8 bit quality sound. Mode 2 offers **2 channels of 16 bit sound** and **2 channels of 8 bit sound.**

You can connect up to four speakers (either amplified speakers, or connected through a hi-fi) to the **Quazar Surround** . If only two speakers are connected then the sound is played as stereo, but if four speakers are connected then the sound is surround, giving total immersion in an audio landscape. The **Quazar Surround** also has an expansion port for a stereo sampler so that sounds can be recorded.

What else? The **Quazar Surround** comes with a full manual and three disks of software.
Introductory Disk – Introductory software demonstrating what the **Quazar Surround** can do, test software, plus a **Quazar Surround** MOD player.
Intro Disk 2: Utilities – Software to convert samples from other computers into **Quazar Surround** format, and easy to use sample players for use in BASIC and Machine Code – which you are free to use in your own programs.
Soundbyte – A free issue of the regular **Quazar Surround** support disk, featuring games, demos and utilities only for the **Quazar Surround** .

The QUAZAR SURROUND *Soundcard*

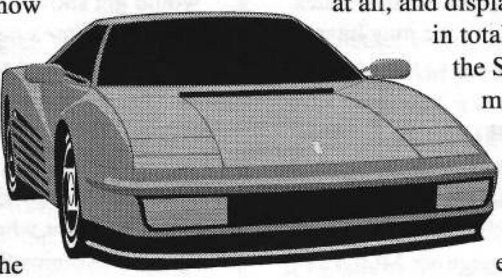
Only £53.99

For more information on Quazar products, including the **Quazar Surround Sampler Module**, send an S.A.E. to the above address.

Speed Up Your Code!

As all good programmers know, timing is everything. From the simple loop that needs optimising to that tricky sample tracker that needs things just so, you need to know how much time things take. And so, armed with just SAM and television, I decided to take yet another look at just how long each machine code instruction takes to execute, and this time to get it right.

The very first thing we need to know is how fast the Z80B really is in the SAM. Looki in any manual and it will tell you 6Mhz, and if you still don't believe it you can open up the case and see a nice little 24Mhz crystal connected to the ASIC. Admittedly this isn't a 6MHz crystal, but this doesn't matter, because we know that the ASIC merely divides this by four and feeds it to the Z80B. Since crystals are fairly accurate things to say the least, we can confidently say that the Z80B operates at 6 million clock cycles per second, give or take at most a few hundred.



Picture Frame

Now that we know how fast the processor is running, we can go on to find how many clock cycles there are in a single frame. This is a very important calculation, as it gives us the number of t-states in a frame. To do this, we'd better figure out what one frame actually is.

A television picture is made up of 625 scan lines, all of which are displayed in 1/25th of a second. These lines are interlaced, with each frame (forming alternate lines of the picture) taking 1/50th of a second, in which time 312½ scan lines must have been displayed. The SAM does not interlace at all, and displays 312 scan lines in total, so each frame the SAM produces must take just under 1/50th of a second (0.019968 sec, to be precise). It is easy to show that exactly 119,808 clock cycles have occured in this time, which gives us 119,808 t-states to play with.

Under the Microscope

That's about as far as theory will take us for the moment, so it's time to perform some tests on the SAM itself. What we want to do is repeatedly run through a single instruction of known length between two successive frame interrupts, and see how many were executed. NOP is generally chosen for this purpose, because it's one of the shortest instructions to execute, and doesn't actually do anything, making it a nice standard reference.

Unsurprisingly, the results do not match the theory, and after some fiddling around with different screen modes, we come up with the following results, if we take NOP to be exactly 4 t-states long:

THEORETICAL	119,808 t-states
SCREEN OFF	119,696 t-states
MODES 2-4	95,120 t-states
MODE 1	77,468 t-states

So what's happening to all the missing t-states? The processor clock speed can't have changed, so something must be interfering with the execution time of the instructions, giving the illusion of a reduction in the number of t-states in a frame. It looks like we may have to add to our theory.

Point of Contention

One of the main things to notice is that there is exactly a 24,576 t-state difference between the screen display being on or off, if we ignore MODE 1 for the moment because the processor is deliberately slowed down in this mode to help ZX Spectrum software to run at nearly the correct speed. As the screen is displayed during a frame, the ASIC needs to access memory to find out what to display, and we can assume that it is this memory contention that is wasting time. In fact, it corresponds to 1 t-state for every byte of screen memory in MODEs 3 & 4. Looking at this a little more closely, and with some careful programming, we find something fundamental about the operation of the SAM.

What we find is that far from 1 t-state apparently being lost for every byte of screen memory, we have 4 t-states worth of execution lost for every 4 bytes of screen memory in MODEs 3 & 4, and that we can only execute instructions within 4 t-state boundaries. What is happening is that the ASIC is pipelining 4 bytes of screen memory at a time, ready to be displayed (memory chips are designed to parcel out data quickly in this way if a circuit can take advantage of it), and since the Z80B can only access the internal 512K through the ASIC, it is forcing the Z80B to wait while it gets priority access to the memory. If the ASIC allowed the Z80B to have priority access to the memory, then we would get snow – break up on the screen *(Editor's note: this particular effect could be seen on the prototype ASIC – see the article on p.34 of this issue)*. However, the ASIC also issues WAIT states to the Z80B to keep it in synchronisation with its burst reads at all times, even when it doesn't need to, and this explains the 4 t-state boundaries.

The only time that this doesn't occur is when using external memory, which is accessed directly (with a little help for the paging), and the ROM. In these cases, you can refer back to any standard list of Z80 execution times, and you don't really need to read any further. Not that many people own 1Mb expansions, but having said that, I'd recommend using it whenever possible - it'll make your code a lot faster as long as it doesn't access internal memory.

Working back, we can see that with the screen off, we have 312 lines, each 384 t-states long. In MODEs 2-4, however, 192 lines now constitute the screen area, with 128 t-states in the border area of each line, and 256 t-states in the screen area, of which 128 t-states are in contention.

There is, however, a way to avoid some memory contention, if not the 4 t-state synchronisation, without resorting to using a megabyte expansion. If we can find opcodes that do not need to access memory while the ASIC is doing so, the instruction will continue over the contended period without waiting. These opcodes do exist, and we now know enough to start looking for them and to time every instruction.

If we time each instruction in the border area, or with the screen off, we know that there is no memory contention and we find the length of the instruction. This can either be done by using a frame interrupt and running the instruction repeatedly as we did for NOP, or by changing the colour look-up table to get a horizontal bar on-screen whose length matches the execution time of a single instruction.

To find those instructions that do not access memory for a period during their operation, this time we change the colour look-up table again, but this time we make sure that the instruction is executing during memory contention, in other words in the screen area.

Nomenclature

n	8 bit number
s	8 bit signed number
nn	16 bit number
r	8 bit register
rr	16 bit register
IXh	8 bit index register
IX	16 bit index register
b	bit
c	condition
i	interrupt mode.
comp	compound instruction - any bit or shift/rotate operation, including SLL.
/	condition not met / condition met.
(+4)	4 additional t-states to access an ASIC port. For external memory, time must first be rounded up to nearest 4 t-states.
*	does not access memory over one 4 t-state sync.
**	does not access memory over two 4 t-state syncs.

Prefixes

u	undocumented instruction that is (probably) accelerator compatible.
U	undocumented instruction which is unlikely to be compatible with any future accelerator.
D	undocumented duplication of existing instruction - incomplete decoding is responsible for these instructions. They are also unlikely to be compatible with any future accelerator.
+	The R register is unlikely to be used for refresh on an accelerator, in other words don't expect the value to increment

with time.

m Values quoted for fully internal or fully external memory operation (i.e. data source/destination and code location).

Instruction Timings

	Mnemonic	Int	Ext
	ADC A,r	4	4
	ADC A,n	8	7
	ADC A,(HL)	8	7
u	ADC A,IXh	8	8
	ADC HL,rr	16*	15
	ADC A,(IX+n)	20*	19
	ADD A,r	4	4
	ADD A,n	8	7
	ADD A,(HL)	8	7
u	ADD A,IXh	8	8
	ADD HL,rr	12*	11
	ADD IX,rr	16*	15
	ADD A,(IX+n)	20*	19
	AND r	4	4
	AND n	8	7
	AND (HL)	8	7
u	AND IXh	8	8
	AND (IX+n)	20*	19
	BIT b,r	8	8
	BIT b,(HL)	12	12
	BIT b,(IX+n)	24*	20
	CALL nn	20	17
	CALL c,nn	12/20	10/17
	CP r	4	4
	CP n	8	7
	CP (HL)	8	7
u	CP IXh	8	8
	CP (IX+n)	20*	19
	CCF	4	4
m	CPD	16*	16
m	CPDR	16*/20*	16/21
m	CPI	16*	16
m	CPIR	16*/20*	16/21
	CPL	4	4
	DAA	4	4
	DEC r	4	4
	DEC rr	8*	6
u	DEC IXh	8	8
	DEC IX	12*	10
	DEC (HL)	12	11
	DEC (IX+n)	24*	23
	DI	4	4
	DJNZ s	12*/16**	8/13
	EI	4	4
	EX rr,rr	4	4
	EX (SP),HL	20	19
	EX (SP),IX	24	23
	EXX	4	4
	HALT	4	4
	IM i	8	8
D	IM i	8	8
U	IM 0*	8	8
D	IM 0*	8	8
	IN r,(C)	12(+4)*	12(+4)
U	IN X,(C)	12(+4)*	12(+4)
	IN A,(n)	12(+4)*	11(+4)
m	IND	16(+4)*	16
m	INDR	16(+4)*/ 20(+4)*	16/ 21
m	INI	16(+4)*	16
m	INIR	16(+4)*/ 20(+4)*	16/ 21
	INC r	4	4
	INC rr	8*	6

u	INC IXh	8	8
	INC IX	12*	10
	INC (HL)	12	11
	INC (IX+n)	24*	23
	JP (HL)	4	4
	JP (IX)	8	8
	JP nn	12	10
	JP c,nn	12	10
	JR s	12*	12
	JR c,s	8/12*	7/12
	LD r,r	4	4
	LD r,n	8	7
	LD r,(rr)	8	7
u	LD r,IXh	8	8
	LD A,I	12*	9
+	LD A,R	12*	9
	LD I,A	12*	9
	LD R,A	12*	9
	LD A,(nn)	16	13
	LD A,(IX+n)	20*	19
U	LD r,comp(IX+n)	24	23
u	LD IXh,r	8	8
u	LD IXh,IXh	8	8
u	LD IXh,n	12*	11
	LD (rr),A	8	7
	LD (HL),r	8	7
	LD (HL),n	12	10
	LD (nn),A	16	13
	LD (IX+n),r	20*	19
	LD (IX+n),n	24*	19
	LD rr,nn	12	10
	LD IX,nn	16	14
	LD SP,HL	8*	6
	LD SP,IX	12*	10
	LD HL,(nn)	20	16
	LD rr,(nn)	24	20
	LD IX,(nn)	24	20
	LD (nn),HL	20	16
	LD (nn),rr	24	20
	LD (nn),IX	24	20

m	LDD	20	16
m	LDDR	20/24	16/21
m	LDI	20	16
m	LDIR	20/24	16/21
	NEG	8	8
D	NEG	8	8
	NOP	4	4
	OR r	4	4
	OR n	8	7
	OR (HL)	8	7
u	OR IXh	8	8
	OR (IX+n)	20*	19
	OUT (n),A	12(+4)*	11(+4)
	OUT (C),r	12(+4)*	12(+4)
U	OUT (C),0	12(+4)*	12(+4)
m	OUTD	16(+4)*	16
m	OTDR	16(+4)*/	16/
		20(+4)*	21
m	OUTI	16(+4)*	16
m	OTIR	16(+4)*/	16/
		20(+4)*	21
	POP rr	12	10
	POP IX	16	14
	PUSH rr	12	11
	PUSH IX	16	15
	RES b,r	8	8
	RES b,(HL)	16	15
	RES b,(IX+n)	24	23
	RET	12	10
	RET c	8*/12	5/11
	RETI	16	14
D	RETI	16	14
	RETN	16	14
D	RETN	16	14

	RL r	8	8
	RL (HL)	16	15
	RL (IX+n)	24	23
	RLA	4	4
	RLCA	4	4
	RLC r	8	8
	RLC (HL)	16	15
	RLC (IX+n)	24	23
	RLD	20*	18
	RR r	8	8
	RR (HL)		
		16	
			15
	RR (IX+n)	24	23
	RRA	4	4
	RRC r	8	8
	RRC (HL)	16	15
	RRC (IX+n)	24	23
	RRCA	4	4
	RRD	20*	18
	RST n	12	11
	SBC A,r	4	4
	SBC A,n	8	7
	SBC A,(HL)	8	7
	SBC A,(IX+n)	20*	19
u	SBC A,IXh	8	8
	SBC HL,rr	16*	15
u	SBC A,IXh	8	8
	SCF	4	4
	SET b,r	8	8
	SET b,(HL)	16	15
	SET b,(IX+n)	24	23
	SLA r	8	8
U	SLL r	8	8
U	SLL (HL)	16	15
U	SLL (IX+n)	24	23
	SRA r	8	8
	SRA (HL)	16	15
	SRA (IX+n)	24	23
	SRL r	8	8
	SRL (HL)	16	15
	SRL (IX+n)	24	23
	SUB r	4	4
	SUB n	8	7
	SUB (HL)	8	7
u	SUB IXh	8	8
	SUB (IX+n)	20*	19
	XOR r	4	4
	XOR n	8	7
	XOR (HL)	8	7
u	XOR IXh	8	8
	XOR (IX+n)	20*	19

The Rest Is Up To You...

So what does all this mean? Well, with the screen off or when updating in the border area, just take the numbers given, and that's the number of t-states it takes to execute the instruction. If, on the other hand, the instruction executes during a period of memory contention (i.e. during screen update), then the instruction will take twice as long to complete, waiting for 4 t-states while the ASIC accesses memory for every 4 t-states it takes to execute. For every asterisk next to the execution time, however, you can take 8 t-states off the final execution time since the instruction has managed to continue without accessing the memory while the ASIC isn't ready. The total number of t-states in a frame remains fixed at 119,808, at all times.

David Zambonini

NSSS

The Northern SAM & Spectrum Show

Saturday 21st September 1996

Most of the major SAM and Spectrum companies will be here, so we want you to be as well! Such leading lights as **Alchemist Research, Fountain PD, FRED Publishing, FORMAT, Persona, SAM PD** and many many more will be there for you to peruse their wares, as well as a large collection of SAM and Spectrum personalities. If you can't face the jaunt down to Gloucester, then why not come to sunny Leeds? It's only **£1.50** on the door for entry to the Northern SAM and Spectrum Show, but with this magazine you'll find some money-off coupons which will allow you into the building for only ***£1.00!*** You'd be crazy to miss an opportunity like this!

The show is being held in the **Methodist Hall** just near **Westgate**, on **Church Street** in Wetherby. If you get lost, the local Tourist Information bureau should be able to point you in the right direction!

Getting There

By Train

There isn't a train station in Wetherby, but buses run from nearby Leeds. Bus numbers **98** and **99** run from near to Leeds City Train Station on **Infirmary Street** (walk over City Square and the bus stop is outside the Post Office). They run every half hour, at 5 minutes past and 25 minutes to the hour. The cost is approx. 60p, and it'll take about 45 minutes. The first bus runs at 6:05am.

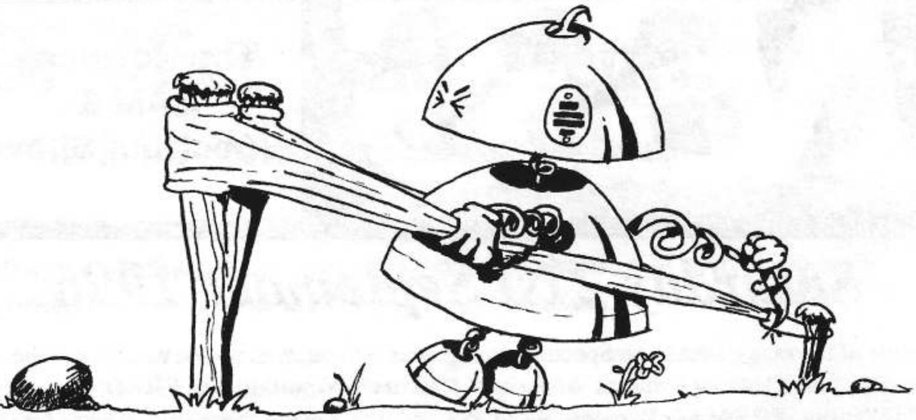
By Coach

There is a *National Express* station in Wetherby, and you should be able to find your way to the show quite easily – the coach station is just down the road from the hall. Again, ask Tourist Information if you get lost. If you can't get a coach to Wetherby, then get one to Leeds. Once at the coach station, just walk through to the bus station (they're part of the same building), and catch the **760** to Wetherby. It runs on the hour and half-past the hour, starting at 8am, and will cost you about 60p.

By Car

If you can find the A1, you're sorted. *(Sorry, but we've not got the room to print a map here).* Wetherby is mid-way between Leeds and York, and major roads entering Wetherby include the A661, A1 and A58. The A59 just scoots past to the north.

NSSS, 123 Potternewton Lane, Chapel Allerton, Leeds, LS7 3LW • Tel: (0113) 237 4800 • Fax: (0113) 237 4349



The SAM Accelerator

Last issue I discussed all the pro's and con's of the Zilog Z380 CPU as a possible candidate for the SAM Accelerator. Yes, you're right – there were more pros than cons. This is simply because I really like the chip. This issue I will run through the ideas behind the SAM accelerator and explain they could be used to build one.

The accelerator concept is due to Simon Cooke who thrashed out his ideas with Bruce Gordon at a computer show *(Argh! Rumbled! **Simon**)*. The decision was that it was feasible in theory but with problems in practice. The bulk of the non processor housework is carried out by the ASIC which stays in step with the Z80 by issuing wait states.

Point of Contention

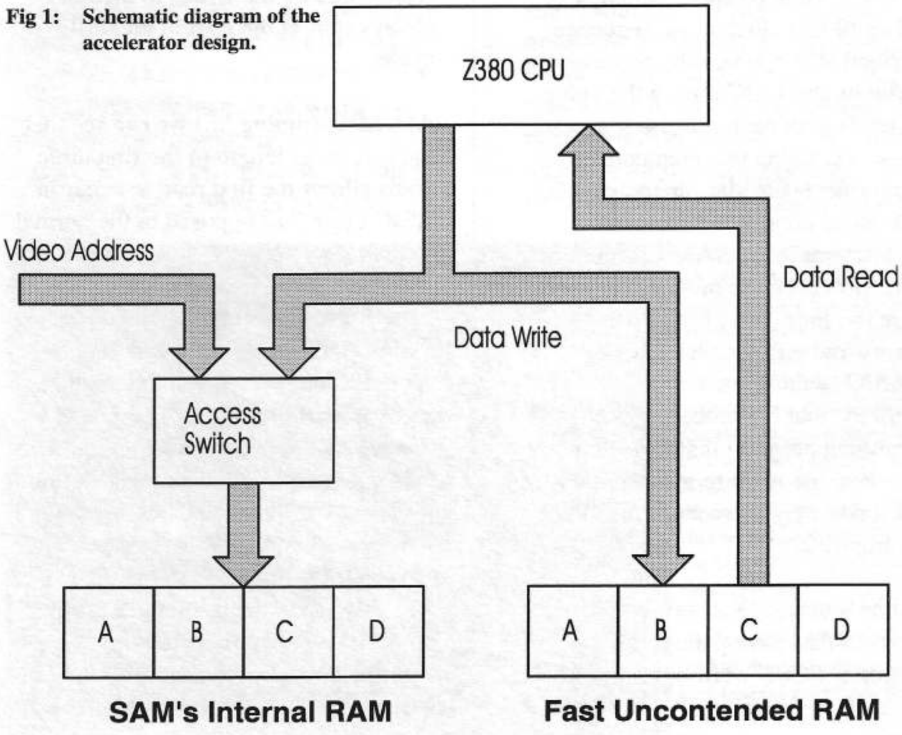
One of the nice things about the SAM's memory is that any part of it can be used for video display with easy switching between one section of memory and another. The drawback of this arrangement is that there is a continual fight between the processor and the video circuitry for access to the memory. If the processor was given priority over the video circuits then there would be random snow all over the screen rather like a badly tuned TV. Whenever the processor accessed memory the video circuits would not have access to the information it needs to display and hence, would be forced to display incorrect colours for the duration of the access. If, however, the video circuitry had priority over the processor then the processor could only access during the line synchronizing periods of the screen. Each screen line lasts for 64 μs (micro seconds) of which 56 μs are used to scan the viewable part of the picture. This means that the processor could only access the program memory for 12.5% of the time which for a 6MHz Z80B would mean an effective execution speed equal to 0.75 MHz.

Your Flexible Friend

Clearly a better access system is required and the one used by Bruce in the SAM is pretty good – see your nearest SAM for the proof. The ASIC works on a four CPU clock cycle sequence and uses a squirrel store type of reservoir thingy system. Basically the ASIC allows the processor to access the memory at the start of the sequence and once the processor has finished its read or write the ASIC then reads quite a few bytes from the memory sequentially at a speed of several bytes per micro second. These reads are at a rate far greater than the bytes that are used for the screen display; so the excess bytes are stored and used to maintain the video display during the early part of the next sequence when the processor is given memory access again. If the processor attempts another memory access during the video burst read period, the ASIC asserts the WAIT(L) line to the processor causing it to pause until the ASIC is ready to release the memory at the end of the sequence and de-asserts the WAIT(L) line. The processor then carries on as normal. The trade off is that the processor is sometimes kept waiting while the video circuitry does its bit, with the result that the Z80B has an effective speed of about 4.4 MHz.

One thing that Simon realized is that most memory accesses by the processor are READs and that a read does not change or update anything in the memory. Hence the main concept

Fig 1: Schematic diagram of the accelerator design.



behind the accelerator is to have a second set of memory which is not contended by the video memory; i.e. only the processor can access it. His idea is that when the processor writes data to the memory it is written to both copies of the RAM with the processor being subject to the wait restrictions issued by the ASIC. However, when the processor reads data it only reads it from the uncontended version of the memory where there are no speed restrictions caused by wait states. The writes must write to both memories as the video display system will continue to use the original SAM RAM as its source of display data.

Clock This!

I have heard it said that we could simply put in a faster Z80 and clock it faster. The main problem with that is the four 6MHz clock cycle sequence described above where the processor is held up by the ASIC during the video data reads after each processor access to memory. Using the uncontended memory for reads idea, an increase in clock speed does make some improvement in the SAMs execution speed. However, the processor will, before too long, need to write to memory and will then have to wait for the ASIC again. If we assume, for simplicity, that the processor is executing a program that does three reads then one write to memory then three reads etc., the scenario would be as in Figure 2.

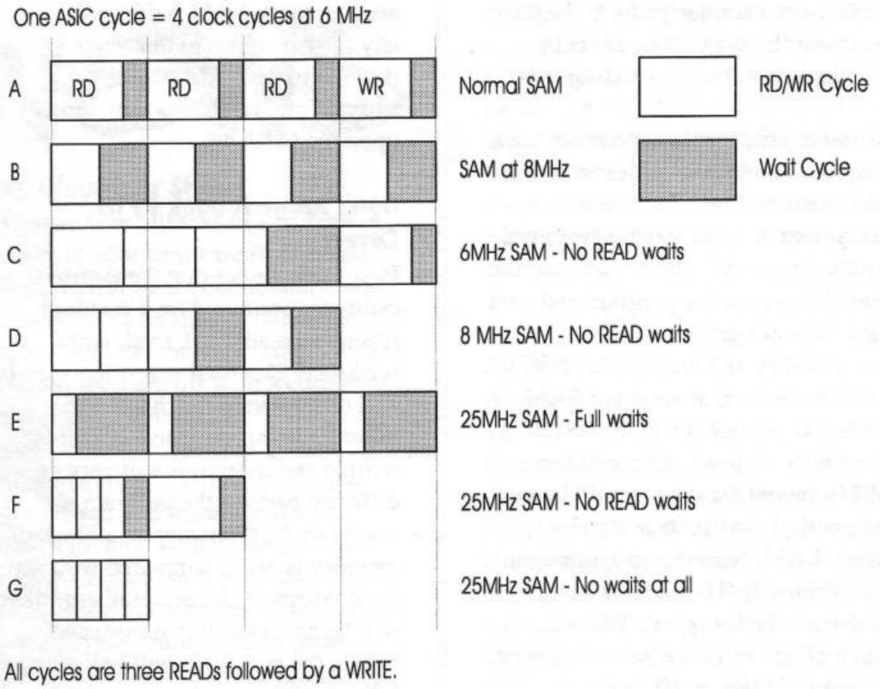
The line labelled 'Normal' is a standard SAM timing diagram showing the CPU waiting for the ASIC after each read/write cycle. If we now compare this with timing 'b' (8MHz Clock with full waits) we can see that the waits slow the CPU down to the same effective execution speed as the 6MHz SAM.

If we now implement the shadow RAM which gives us memory READs with no wait states (timing line 'c'), we can see that we can reduce the read, read, read, write sequence by one ASIC cycle which gives us an effective execution speed of approximately 5.3 MHz. This is due to the read immediately after the first write starts one clock cycle earlier, which has an accumulative knock on effect later where there is no wait before the second write. Hence, this means that the third sequence starts 4 clock cycles earlier and so on ... Notice that all the writes to memory always start at the start of an ASIC cycle.

At 8 MHz (timing 'c') we can see that the shortened length of the first three reads allows the first read to occur in ASIC cycle 3 as opposed to the normal cycle 4.

Even More Problems...

As the clock speed increases we hit another timing problem – the memory access speed of the ASIC. The RAM in the SAM is DRAM which requires a multiplexed address which takes time for the ASIC to set up. If we try writing to the DRAM before the ASIC has loaded the address into the DRAMS using the RAS and CAS lines then the data can be written to undefined random bytes. This could be seen at the Gloucester show where the


One ASIC cycle = 4 clock cycles at 6 MHz
A
RD
RD
RD
WR
B
C
D
E
F
G
Normal SAM
SAM at 8MHz
6MHz SAM - No READ waits
8 MHz SAM - No READ waits
25MHz SAM - Full waits
25MHz SAM - No READ waits
25MHz SAM - No waits at all
RD/WR Cycle
Wait Cycle
All cycles are three READs followed by a WRITE.

demo accelerator had a tendency to write data indiscriminately all over the screen. In actual fact this effect was corrupting the whole of the SAMs internal memory but fortunately the accelerator runs from the program stored in the external shadow static RAM which *can* keep pace with the CPU with no effort.

It is this access time problem which causes the writes to be longer in duration than the reads on the faster timing waveforms. Waveform 'e' shows this effect at 25MHz.

A solution to this problem, which has yet to be prototyped, is to place a buffer between the accelerator and the SAM. This buffers job would be to accept memory writes from the accelerator at top speed and then emulate a 6MHz Z80 to write to the SAMs memory while being responsive to the ASICs wait states. see waveform 'g'.

One thing to note about the buffer is that it can accept one write only for each ASIC cycle. This means that no matter how short the reads become the cpu is still forced to wait for the next ASIC cycle.

Speed Me Up Scotty!

Last issue I ranted about the Z380 and its possible use in the accelerator. The Z380 has internal pipelining, which means that any instruction will take 2 'T states' only. Also, the read-write cycles take only 2 clock cycles. This means an increase in execution speed of 50% over a similarly clocked Z80. Waveform 'h' shows that this is not advantageous at higher clock speeds.

Do not despair, I have yet another trick up my virtual electronic sleeve. Not all of the memory is used for screen display, two or three screens is typical for most programs. This means that the remainder is used for program and data which does not get accessed by the video circuitry and hence, does not need to be written to the main SAM memory. If we select which parts of the memory map are written to the SAM (screens) then the remainder can reside entirely in the uncontended shadow RAM. Now our programs can run at a true 25MHz Z380 (when they become available) speed. This will give an effective Z80 equivalent speed of around 37MHz or 700% !!!!

This leaves only one problem remaining; 'How do we decide what is screen data and what is not?' This will be taken care of using a special bit of circuitry which will cause an interrupt when the screen register VMPR is changed. When this interrupt occurs an interrupt servicing routine will check to see if the new page has been used as a screen before. If it has it simply returns to the user program; if not it will then designate that part of the memory map as requiring write to the SAM memory and then read each byte of that block of memory and re-write it back to the same address. This will have the effect that the data is read from the shadow RAM and written back to the same address in both the shadow and the SAM memory. From that point on (until told otherwise) the accelerator to SAM buffer will copy any further writes to that screen through to the SAM with the appropriate wait states now forced upon the CPU.

Don't Judge A Book By Its Cover

Please remember that these speed estimates are based on a standard repetitive read, read, read, write sequence. The final execution speeds will depend very much upon the program being executed, how it is written and indeed, it will vary as different parts of the program are executed. If, for example, a block of memory is being compared with say the contents of the accumulator, there will be no writes to memory and, hence, the processor will belt along at full speed!

Next issue I shall explain how we can use commonly available PC parts to replace SAM parts which are difficult to come by, such as floppy drive controllers.

Martin Rookyard

We Need

YOU!

We're on the look out for technically minded writers. We can offer you two free issues of the magazine for your article, so what are you waiting for? You don't have to be an electronics wizard or a programming guru – you just need to be willing to write an interesting article on software or hardware techniques. What have you got to lose?

Want to place an advert in this quality magazine? Want to reach other SAM users? In that case, give us a ring on (01942) 886084 – we'll be waiting for your call, and we have incredibly reasonable rates!

Based On An Idea...

POSSIBLY THE BEST TECHNICAL MAGAZINE IN THE WORLD

Back Issues

Issue 1 of ***Based On An Idea...*** is still available at the time of going to press. Featuring in depth articles and reviews such as:

How to be a complete Bursterd

Stefan Drissen, creator of the SAM MOD Player, on the principles behind the sample playback routines he uses.

Of Mice and Men...

Part one of a two part article by Simon Cooke on how to read and use the SAM Mouse.

Is there anybody out there?

Dave Whitmore waxes lyrical about the BBS he runs – the first ever SAM BBS in the world!

SAM Gets Wired...

Where to find the SAM on the World Wide Web, and the rest of the Internet.

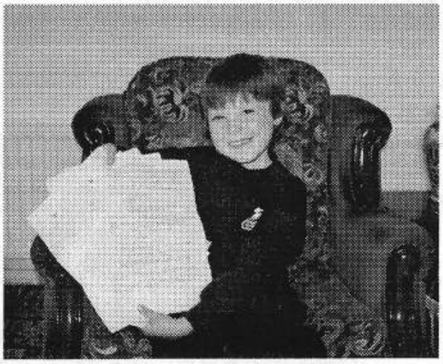
A High Speed SAM?

Martin Rookyard exposes the latest of Zilog's babies with an in-depth look at the Z380 processor with a view to using it in the SAM Accelerator.

Only £2.00 from: ***Based On An Idea,*** **38 Squires Lane, Tyldesley, Manchester, M29 8JF. Cheques made payable to M.Rookyard.**

Prize Winners

In the first issue of *Based On An Idea…* we hid a questionnaire in the middle of the magazine. The people who sent them back to us were put into a draw to win free issues… Were you one of the lucky winners?


Simon Rookyard proudly presents the five winning questionnaires which he drew from the many we received.

The winners of the competition we held in the first issue of the magazine are: A.L.Bennett, David Brant, James Curry Phil Glover and Cliff Jackson, who filled in their questionnaires (well, Phil didn't – he printed out his own version). Those lucky people win two free copies of the magazine. Thanks to everyone who sent theirs in – we couldn't put this magazine together without this kind of feedback, and your efforts are very much appreciated. Keep your eyes peeled for another competition soon!

Available NOW!

Limited stocks so don't delay!*

SAM MOD player version 2.04

- Output frequency of **10.4kHz** (with screen on!)
- play Amiga MODs on your SAM via the soundchip, SAMdac, printer dac, Blue Alpha Sampler or Quazar
- **ALL** Protracker commands implemented so you can hear your MODs how they were meant to be heard! (even BPM changes and the Extended effects are implemented)
- Load MODs **directly from PC or SAM discs** (the file selector also only shows MOD files so you don't need to wade through the directory looking for that MOD file – handy for FRED discs!)
- COMET source file showing you how to use MODs in your own code!

All this for **only FIVE pounds** cash (or Eurocheque for 12.50 dutch gilders). For an extra *£5* (etc) you also get the **FULL COMET SOURCE!!!**

SAMdac

- play MODs at **7 bits per channel in stereo!** (8 times the soundchip's quality – half the quality of the Quazar)
- connects to your printer interface and audio out port (interface has a built-in through port)
- **FREE copy** of the latest version of the **SAM MOD player!**

Sick of listening to your SAM's feeble attempts at reproducing Amiga MODs but not prepared to shell out for the Quazar? Then the **SAMdac** is the answer for you, priced at **only *£25*** cash (or a Eurocheque for 65 dutch guilders)

BOTH products are available from:
Stefan Drissen, Zevende Herven 6, 5232 JZ 's-Hertogenbosch, The Netherlands
Eurocheques made payable to Stefan Drissen

*fortunately, stocks are not limited – sounds spectacular though, doesn't it?

Contacts!

All postal addresses are in the UK unless otherwise noted, as are all phone numbers. If you're dialling from outside the UK, dial your international dialling code, followed by **+44**, then the UK number (minus the leading zero). Don't forget to include an SAE with your correspondance if you want a reply.

Software

Atomik Software
⊠ Dept. BOAI, 20 Grove Road, Hoylake, Wirral, Merseyside, L47 2DT

B.G. Services
⊠ 64 Roebuck Road, Chessington, Surrey, KT9 1JX *(please write all letters / orders clearly)*
☎ (0181) 2874180

Elysium Software (SAM Adventures)
⊠ 50 Chadswell Hgts, Lichfield, Staffs, WS13 6BH

F9 Software
⊠ 18 Mill Lane, Glenburn Rd, Skelmersdale, Lancs, WN8 8RH
☎ (01695) 31163

FRED Publishing
⊠ 40 Roundyhill, Monifieth, Dundee, DD5 4RZ
☎ (01382) 535963

Hilton Computer Services Ltd
⊠ 3 Suffolk Drive, Guildford, Surrey, GU4 7FD
☎ (01483) 578983

Jupiter Software
⊠ 2 Oswald Rd, Rushden, Northants, NN10 0LE

Persona
⊠ 31 Ashwood Drive, Brandlesholme, Bury, Lancs, BL8 1HF
☎ (0161) 797 0651

Revelation Software
⊠ 45 Buddle Lane, Exeter, EX4 1JS

Saturn Software
⊠ 5 Ivanhoe Drive, Westfields, Ashby de la Zouch, Leics., LE65 2LT

S.D. Software
⊠ 70 Rainhall Road, Barnoldswick, Lancashire, BB8 6AB

Steve's Software
⊠ 7 Narrow Close, Histon, Cambridge, CB4 4XX
☎ (01223) 235150 from 6pm-9pm, Mon-Sat

Supplement Software
⊠ 37 Parker St, Bloxwich, Walsall, WS3 2LE
☎ (01922) 406 239

Hardware

B.G. Services (see under Software)

Colin Piggot
⊠ 204 Lamond Drive, St. Andrews, Fife, KY16 8RR

Rooksoft
⊠ 38 Squires Lane, Tyldesley, Manchester, M29 8JF
☎ (01942) 886084

West Coast Computers
⊠ see *FORMAT Publications.*

Publications

Adventure Probe
⊠ Barbara Gibb (Editor), Adventure Probe, 52 Burford Road, Liverpool, L16 6AQ

Based On An Idea...
⊠ 38 Squires Lane, Tyldesley, Manchester, M29 8JF (or email us at entropy@jumper.mcc.ac.uk)
☎ (01942) 886084

Crashed
⊠ 16 The Avenue, Manston, Leeds, LS15 8JN
☎ (0113) 232 6726

FRED Disk Magazine
⊠ See *FRED Publishing.*

FORMAT Publications
⊠ 34 Bourton Road, Gloucester, GL4 0LE
☎ (01452) 412572
📠 (01452) 380890

Zodiac
⊠ New House, Holbear, Chard, Somerset, TA20 2HS
☎ (01460) 62118

Books & Manuals

B.G. Services (see under Software) for genuine Digital Research CP/M 2.2 manuals

FORMAT Publications (see under software) for SCADS manuals

Demo Groups

Entropy
⊠ 18 Braemar Drive, Sale, Cheshire, M33 4NJ (email: entropy@jumper.mcc.ac.uk)

MNEMOtech
⊠ MNEMOtech, c/o Andrew Collier, 57 Wyndham Avenue, Bolton, BL3 4LG
☎ (01204) 652470

⊠ Postal address (or email address)
☎ Telephone number
📠 Fax number

Next Issue

- **The Butterfly Print: The fastest, flexible, colour, Mode 3, 85-column character printing routine ever devised**
- **SAM Comms Special – how to modify your Comms Interface so that it works**
- **News, Letters, our very own help page and much much more... –** ***BUT ONLY IF YOU WRITE IN!***

Available from the end of October £2.00 per issue, or see our subscription offers!

Index of Advertisers

IFC	Rooksoft, 38 Squires Lane, Tyldesley, Manchester, M29 8JF, UK
6	SAM PD, 18 Mill Lane, Glenburn Rd, Skelmersdale, Lancs, WN8 8RH, UK
26	Quazar, Colin Piggot, 204 Lamond Drive, St. Andrews, Fife, KY16 8RR, UK
33	NSSS, c/o Allan Clarkson,
40	Stefan Drissen, Zevende Herven 6, 5232 JZ 's-Hertogenbosch, The Netherlands
IBC	FRED Publishing, 40 Roundyhill, Monifieth, Dundee, DD5 4RZ
BC	Persona, 31 Ashwood Drive, Brandlesholme, Bury, Lancs, BL8 1HF, UK

IFC = Inside Front Cover, IBC = Inside Back Cover, BC = Back Cover

Editors: Simon Cooke, Maria Rookyard, Martin Rookyard • **DTP:** Simon Cooke • **Contributors:** Simon Cooke, Stefan Drissen, Maria Rookyard, Martin Rookyard, David Zambonini • **Text proof reading:** Maria Rookyard • **Technical proof reading:** Simon Cooke, Martin Rookyard • **Illustrations cover art and photography:** Simon Cooke • **Tel:** (01942) 886084 • **Address:** 38 Squires Lane, Tyldesley, Manchester, M29 8JF • **Email:** entropy@jumper.mcc.ac.uk • **WWW:** http://jumper.mcc.ac.uk/~simonc/boai/ • **Subscriptions to** Maria Rookyard at the above address – cheques made payable to M. Rookyard please! • **Special Offer:** Take out a year's subscripton – that's 4 issues of the magazine – for only £6.00! • **Published by** Lame Ducks International Press • **Printed by** Manchester Computing Centre • **This issue** put together to the sounds of the combined talents of David Bowie, Pulp and Tori Amos • ***Based On An Idea...*** is a quarterly magazine from those talented bods at Entropy and Rooksoft. All submissions will be taken under consideration. Contact us about our advertising rates.

Do You Have Artistic Leanings?

FRED

PUBLISHING

Presents two utilities for SAM owners of the *artistic* persuasion

SAM PAINT

The astounding paint package which beats seven shades (or even 16 colours) out of 16 bit packages! Includes perspective, plasma and gradient fill and image deformation options with which you can create an amazing array of graphical effects.

£25.00

SAM SPRITE

Create sprites for use with SCADS, GamesMaster and other game creation utilities! Includes comprehensive animation and screen grabbing features, mouse control and more!

£10.00

FRED Publishing, 40 Roundyhill, Monifieth, Dundee, DD5 4RZ
Tel: 01382 535963 (Evenings and weekends only) Fax: 01382 535622
Cheques made payable to FRED Publishing

BLITZ

The New Bi-Monthly Disk Magazine From

PERSONA

Persona, 31 Ashwood Drive, Brandlesholme, Bury, Lancs, BL8 1HF. Tel: (0161) 797 0651.

Blitz Disk Magazine is £2.00 per issue. Cheques made payable to M.D. MacKenzie. We could tell you what was in it, but then we'd have to kill you…