Frobnicate

SEPTEMBER/
OCTOBER 1995

Issue 4. The ULTIMATE techie magazine for Acorn Enthusiasts.

SPECIAL!!!
DOUBLE ISSUE

© opyright

In this issue:

- ☞ How floppydiscs/harddiscs work.
- ☞ Know your SysOp's for the Acorn Show.
- ☞ About software copyright.
- ☞ Simple Centronics interface.
- ☞ Ants!
- ☞ And much much more.....

Published by Your Printer

Index:

Distribution stuff:

Editor . . . . . Richard Murray
Contributors . . Richard Murray, Niall Douglas, James Larcombe, Nava Whiteford.
Graphics . . . . Richard Murray, Chris Jackson, Dane Koekoek, whoever did those Acorn Show ’94 pictures...

Editor's Pages

Prepared in Ille-Et-Vilaine, France.

Again, a late Frobnicate. Apologies to all of you readers. You see, we went to France for about 10 days to get the roof reslated and well… The roofer never turned up. The car bust. The AA (Europe) ordered the *wrong* part all the way from Japan. Then the delay in getting a roofer. Now things are running kinda to schedule – 20 days later. :-(

The dollar has been devalued, I'm now a firm believer in "Murphy's Law" and Mum is sitting on the bed, hands around head, rocking back and fro singing, "*I'm tired and I wanna go home*". :-)

On a different note (probably a B#), this issue of Frobnicate is bigger than normal to make up for missing a month. A kind of "double issue". It covers such useful topics as "Knowing your SysOp's for the Acorn World Show", "Software copyright", and "Buying a 2nd user system".

There have been some rather interesting requests via the Reader Survey…

Dump Easy peeC and do some assembler.
Do an article on the Reader's fave Web sites.
Add a comments section.
Use the genesis browser or summat.
How to get a reply out of Acorn ;-)
Latest from those who break Acorn/ARM NDA's (non-disclosure agreement) :-)
Like the Moxon interview, but more interesting people.

Okay…
Easy peeC wasn't overly popular. About 35% so I'm going to try some deeper C stuff. I'd like to do a bit on assembler, but myself I can just manage to remember the old "MOV R15,R14"... I don't have any real Internet access, so I can't surf the Web for you (though people tell me that **http://www.oeh.uni-linz.ac.at:8001/~chris/HATE/hate.html** is worth a visit... There is now a comments page, page 39 in this issue. This magazine was originally designed to be a magazine. The idea is you print it. Besides, I didn't find Genesis to be really suitable for something like this... As for getting a reply from Acorn, I'm still waiting for official permission to use SWI chunk &16F00. I've been waiting now a year and a half. I've decided to use it anyway. It is in the system area, but I've not found anything that comes even close to &16F00. Anybody that gets a quick reply from Acorn - let us all know! :-) Right. The NDAs. Well, I don't know anybody that has been daft enough to break an NDA - but if somebody is out there, and you have and interesting story... Let us know. Also, if you wish to break an NDA and tell us all about RiscOS3.80 then netmail me!

{ for the uninitiated, there was a momentary scare on the BBS scene a while back when RiscOS 3.80 was uploaded. It turned out to be mangled bits of the RiscOS 3.22 patches and a textfile claiming:

RiscOS 3.80

The supplied module is release A.

Basic features of RiscOS 3.80:

- Enhanced 32 bit interface support for Risc PC
- Maximum hard disc partition size is now two gigabytes
- Up to 8 devices can be attached to the filer
- Enhanced serial port
- Vastly expanded Edit application, based on Zap {not included here}
- BASIC program compiler {not included here}
- Pseudo ARM 6/7 interpreter for ARM 3 machines
- WIMP is cooperative and pre-emptive
- Fully multitasking printer support {not included here}
- Enhanced hypertext interactive help in Windows style
- MPEG routines built in {not included here}

This product requires at least Risc OS 3.10 and 4Mb RAM to patch. Your system must be properly configured, but only an idiot would have their system configured so badly that this patch will fail.

This product is copyright Acorn Computers Ltd.
If you are not a member of the Acorn Development team or the Memphis project and you are found with this product, you will be unconditionally prosecuted.

Acorn Ltd. 01 Dec 1994

Ho-hum!

Like the Moxon interview...? Erm... Erm... Anybody consider themselves an interesting interviewee? Netmail me!

It was interesting to notice that nobody much missed the news, of which Frobnicate has none. But one nameless person wants an article on “The plight of the last goose farm in northern Venezuela”. Any takers? :-) :-)

I’d like to apologise to Tim Hill (from compu$erve) who found that little bug with Impression. Okay, for you and all the others - here is what happened:

Chris Jackson created the issue 3 version with Impression Publisher. I asked him to save it in a backwards compatible format because not everybody has publisher. We agreed to target Style users. Chris then saved out the multi-file method (the directory gubbins) as that was what the very first Impression used. Common sense would say that’s right... Only Style didn’t want to know. Nor did Impression II. I then asked Chris to try the single file bit. That worked with Style, losing only the fancy stuff (like the bendy borders). Unfortunately I wandered off to France in the middle of this, and Chris uploaded the multi-file version. Sorry you downloaded the wrong one. You could always upgrade to Impression Publisher..... ;^)

You may, if you downloaded this yourself, have noticed it is not as large as you’d expect. I’ve taken steps to reduce the number of images and increase the number of object based graphics (that means less sprites, more drawfiles!). Images are nice - sure, but they also munch through memory like I’ve got an 8Mb A5000. When Encina is back on-line, I may start a Frobnicate archive area (that’s “ftp site” to net-nerds) and chuck in all the piccies and articles that didn’t make it. Did you know the original draft of the BBS Security article was about 80K? For all intensive purposes it was trimmed to about 35K - less than half of it’s original size.

So now to those thinking of running a BBS. I suppose soon I’ll have to try and make a comparative review of BBS servers - though it won’t be easy.

ArcBBS: The ’official’ review version of ArcBBS is something like version 0.88 - slightly old. And which is ’officially’ for sale? If a new SysOp wanted ArcBBS - do they get 1.63 or 1.64? How do you order it?

Archiboard: Appears to have wandered off into the realms of Econet/AUN to become a kind of LAN-BBS. There is a basic form of multi-media within Archiboard - but that will need a special terminal to see more than regular ANSI.

ArmBBS: Is the up-and-coming ANSI BBS system for Acorn’s. Unfortunately it is still in development stage and I’ve not seen any manuals. I’d need a long chat with Keith to do ArmBBS any justice. Apparently there is a terminal mode, and I’ve just found out Keith has implemented a free-form filebase (you wander around a directory structure as it is on the disc!). It’s ShareWare, and costs a mere £25 pounds to register - that’s nearly lunatic value for money as this is becoming as good as ArcBBS (at about £99 quid).

NewsFlash: is up-on-coming, suddenly for nowhere. It supports ArcBBS doors and ArcBBS codes.

VHost: Is a viewdata system by Gareth Babb. It is quite complex software - indeed I looked into using it - but being viewdata, it’s a rather small market. For you new-to-comms people weaned on ANSI and HTML, Viewdata is kinda like teletext in appearance.

BBS: Is written in BASIC, simple and probably used by nobody.

RSDFS/Immediate: Is an Acorn-only system that offers full multi-media effects (graphics/text/sound) and operates like a RiscOS filer. It is a good idea, but is a little bit on the slow side as 14400bps modems take almost as long to send a sample as to play it! However given some imagination you might be able to write interactive WIMP applications for RSDFS/I.

RiscBBS: Anybody with sense will upgrade to the superior ArmBBS.

And *that* concludes the Editor’s Pages for this issue.

Byeeeeee!

The rise of ASCII (from Baudot/EBCDIC)

Once upon a time, in a galaxy far far away... Well, in 1880 actually, some guy called J.E.M. Baudot invented the ’Baudot Code’, which was the main way of transmitting information via telegraph – right up into the 1950s. This code is comprised of five binary digits. Using a shifting system (like the shift key on a typewriter), the 32 binary combinations could represent 58 distinct characters – six being duplicated to be used in either mode.

Letters		Figures	Letters		Figures
NA	0 0 0 0 0	NA	A	1 0 0 0 0	1
LS	0 0 0 0 1	LS	LF	1 0 0 0 1	LF
FS	0 0 0 1 0	FS	J	1 0 0 1 0	6
ER	0 0 0 1 1	ER	K	1 0 0 1 1	(
Y	0 0 1 0 0	3	U	1 0 1 0 0	4
S	0 0 1 0 1	.	T	1 0 1 0 1	NA
B	0 0 1 1 0	8	C	1 0 1 1 0	9
R	0 0 1 1 1	-	Q	1 0 1 1 1	/
E	0 1 0 0 0	2	CR	1 1 0 0 0	CR
X	0 1 0 0 1	'	Z	1 1 0 0 1	:
G	0 1 0 1 0	7	H	1 1 0 1 0	+
M	0 1 0 1 1	)	L	1 1 0 1 1	=
I	0 1 1 0 0	NA	O	1 1 1 0 0	5
W	0 1 1 0 1	?	V	1 1 1 0 1	'
F	0 1 1 1 0	NA	D	1 1 1 1 0	0
N	0 1 1 1 1	NA	P	1 1 1 1 1	%

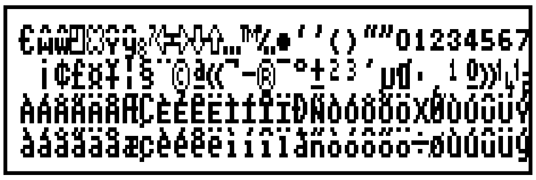
Unfortunately, a single error when transmitting a shift could result in the receiver seeing a stream of garbled stuff: “+5? 1-2 354” instead of “HOW ARE YOU” or “HUU ABA JOU EEAE” instead of “+44 181 654 2212”.

In the 1950s, with the dawn of the computer (yes, it’s only been around 50 years - your parents can remember what it was like before Ninteno), there came a need to represent the lower-case characters as well as special characters (like !@#$&*<> etc) – many more than the 58 provided by the Baudot code. Almost every computer company designed their own way to encode the character set but, not surprisingly, it was IBM’s system that came to be the de-facto standard. This standard was soon revised into the Extended Binary Coded Decimal Interchange Code (EBCDIC). This uses 8 bits to represent 256 unique characters, and could be the origins of the ’8 bits to a byte’ concept.

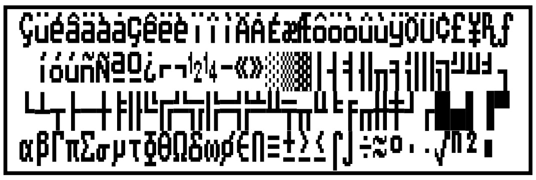
00000000	NULL	01000000	SP	10000000		11000000	
00000001		01000001		10000001	a	11000001	A
00000010		01000010		10000010	b	11000010	B
00000011		01000011		10000011	c	11000011	C
00000100		01000100		10000100	d	11000100	D
00000101	HT	01000101		10000101	e	11000101	E
00000110		01000110		10000110	f	11000110	F
00000111		01000111		10000111	g	11000111	G
00001000		01001000		10001000	h	11001000	H
00001001		01001001		10001001	i	11001001	I
00001010		01001010	£	10001010		11001010	
00001011		01001011	.	10001011		11001011	
00001100		01001100	<	10001100		11001100	
00001101		01001101	(	10001101		11001101	
00001110		01001110	+	10001110		11001110	
00001111		01001111	1	10001111		11001111	CAK
00010000		01010000	&	10010000		11010000	
00010001		01010001		10010001	j	11010001	J
00010010		01010010		10010010	k	11010010	K
00010011	TM	01010011		10010011	l	11010011	L
00010100		01010100		10010100	m	11010100	M
00010101	LF	01010101		10010101	n	11010101	N
00010110	BS	01010110		10010110	o	11010110	O
00010111		01010111		10010111	p	11010111	P
00011000		01011000		10011000	q	11011000	Q
00011001	DC1	01011001		10011001	r	11011001	R
00011010	DC2	01011010	!	10011010		11011010	
00011011	DC3	01011011	$	10011011		11011011	
00011100	STOP	01011100	*	10011100		11011100	
00011101		01011101	)	10011101		11011101	
00011110		01011110	;	10011110		11011110	
00011111		01011111	-	10011111		11011111	DAK
00100000	DS	01100000	-	10100000		11100000	
00100001	SST	01100001	/	10100001		11100001	
00100010	FDS	01100010		10100010	s	11100010	S
00100011		01100011		10100011	t	11100011	T
00100100		01100100		10100100	u	11100100	U
00100101		01100101		10100101	v	11100101	V
00100110		01100110		10100110	w	11100110	W
00100111		01100111		10100111	x	11100111	X
00101000		01101000		10101000	y	11101000	Y
00101001		01101001		10101001	z	11101001	Z
00101010		01101010	<	10101010	SOH	11101010	
00101011	VT	01101011	'	10101011	DLE	11101011	
00101100	FF	01101100	%	10101100	CAN	11101100	
00101101	CR	01101101	-0-	10101101	NAK	11101101	
00101110	SO	01101110	>	10101110	SYN	11101110	
00101111	SI	01101111	?	10101111	ETB	11101111	DOS
00110000		01110000		10110000		11110000	1
00110001		01110001		10110001		11110001	2
00110010		01110010		10110010		11110010	3
00110011		01110011		10110011		11110011	4
00110100		01110100		10110100		11110100	5
00110101		01110101		10110101		11110101	6
00110110		01110110		10110110		11110110	7
00110111		01110111		10110111		11110111	8
00111000		01111000		10111000		11111000	9
00111001	EM	01111001		10111001	STX	11111001	
00111010	SUB	01111010	:	10111010	EXT	11111010	
00111011	ESC	01111011	#	10111011		11111011	
00111100	FS	01111100	@	10111100	EOT	11111100	
00111101	GS	01111101	'	10111101	ENQ	11111101	
00111110	RS	01111110	=	10111110	ACK	11111110	
00111111	US	01111111	"	10111111	BELL	11111111	X

EBCDIC code

In the sixties the ANSI (American National Standards Institute) made an attempt to define a national standard for characters. They did not choose EBCDIC, but instead created ASCII (American Standard Code for Information Interchange). The de-facto ASCII system uses only seven bits to give a standard 128 characters – 32 special codes and 96 alphanumerics. The remaining 128 characters are free for use as graphics characters, parity bits... whatever. However there is no standard for these 128 characters. Below are those 128 characters as an Acorn user will recognise:



Now what a PC user would recognise:



IBMs EBCDIC is still used internally on mainframes, but most microcomputers recognise ASCII.

In BASIC, you can convert to/from ASCII directly by using some of BASIC's functions:

Convert *FROM* ASCII:

`PRINT CHR$(`<asciicode>`)`

so, `PRINT CHR$(65)` would pop up an "`A`" on the screen. An alternative is `VDU 65` – but this isn't suited for programming.

Convert *TO* ASCII:

<variable>`=ASC("`<character>`")`

so, `char%=ASC("A")` would place the value 65 into the variable char%.

You can also convert upper case characters to lower case. This is useful for, say, case insensitive string search routines…

```
IF char%<64 AND char%>91 THEN
  char%=char% OR 32
ENDIF
```

00000000	NUL	01000000	@
00000001	SOH	01000001	A
00000010	STX	01000010	B
00000011	ETX	01000011	C
00000100	EOT	01000100	D
00000101	ENQ	01000101	E
00000110	ACK	01000110	F
00000111	BEL	01000111	G
00001000	BS	01001000	H
00001001	HT	01001001	I
00001010	LF	01001010	J
00001011	VT	01001011	K
00001100	FF	01001100	L
00001101	CR	01001101	M
00001110	SO	01001110	N
00001111	SI	01001111	O
00010000	DLE	01010000	P
00010001	DC1	01010001	Q
00010010	DC2	01010010	R
00010011	DC3	01010011	S
00010100	DC4	01010100	T
00010101	NAK	01010101	U
00010110	SYN	01010110	V
00010111	ETB	01010111	W
00011000	CAN	01011000	X
00011001	EM	01011001	Y
00011010	SUB	01011010	Z
00011011	ESC	01011011	[
00011100	FS	01011100	/
00011101	GS	01011101	]
00011110	RS	01011110	^
00011111	US	01011111	_
00100000	SPC	01100000	`
00100001	!	01100001	a
00100010	"	01100010	b
00100011	#	01100011	c
00100100	$	01100100	d
00100101	%	01100101	e
00100110	&	01100110	f
00100111	'	01100111	g
00101000	(	01101000	h
00101001	)	01101001	i
00101010	*	01101010	j
00101011	+	01101011	k
00101100	,	01101100	l
00101101	-	01101101	m
00101110	.	01101110	n
00101111	/	01101111	o
00110000	0	01110000	p
00110001	1	01110001	q
00110010	2	01110010	r
00110011	3	01110011	s
00110100	4	01110100	t
00110101	5	01110101	u
00110110	6	01110110	v
00110111	7	01110111	w
00111000	8	01111000	x
00111001	9	01111001	y
00111010	:	01111010	z
00111011	;	01111011	{
00111100	<	01111100	|
00111101	=	01111101	}
00111110	>	01111110	~
00111111	?	01111111	DE

ASCII codes

NUL	Null	
SOH	Start Of Heading	Ctrl+A
STX	Start Of Text	Ctrl+B
ETX	End Of Text	Ctrl+C
EOT	End Of Transmission	Ctrl+D
ENQ	Enquiry	Ctrl+E
ACK	Acknowledge	Ctrl+F
BEL	Bell	Ctrl+G
BS	Backspace	Ctrl+H
HT	Horizontal Tab	Ctrl+I
LF	Line Feed	Ctrl+J
VT	Vertical Tab	Ctrl+K
FF	Form Feed	Ctrl+L
CR	Carriage Return	Ctrl+M
SO	Shift Out	Ctrl+N
SI	Shift In	Ctrl+O
DLE	Data Link Escape	Ctrl+P
DC1	Device Control 1	Ctrl+Q
DC2	Device Control 2	Ctrl+R
DC3	Device Control 3	Ctrl+S
DC4	Device Control 4	Ctrl+T
NAK	Negative Acknowledge	Ctrl+U
SYN	Synchronous Idle	Ctrl+V
ETB	End of Transmission Block	Ctrl+W
CAN	Cancel	Ctrl+X
EM	End of Medium	Ctrl+Y
SUB	Substitute	Ctrl+Z
ESC	Escape	
FS	File Separator	
GS	Group Separator	
RS	Record Separator	
US	Unit Separator	
SPC	Space	
DEL	Delete	

Those are the *official* assignments. In real life they do different things, like Ctrl+B on the Acorn will enable the printer...

Another BBS for you to call…

THE WEREWOLF BBS
Online 7pm to 7am

0181 289 6003

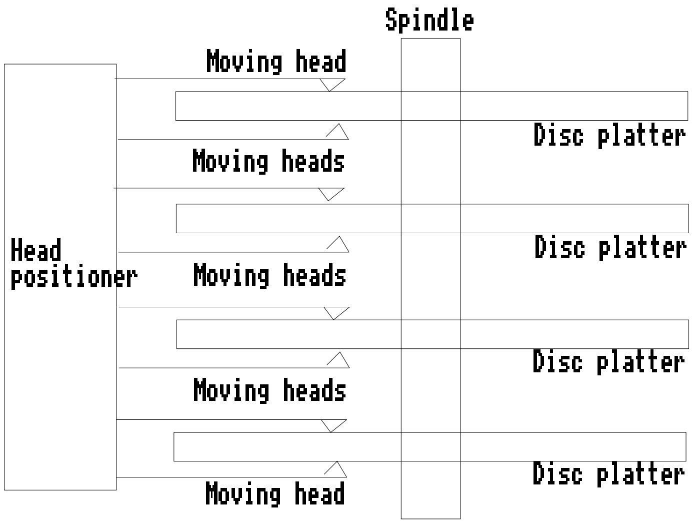
SysOp : Dane Koekoek (that's pronounced "koo-koo" :-))

Hard discs

The hard disc is a mystical thing. Nowadays, for about 200 pounds, you can buy an IDE harddisc capable of storing 1Gb of data (that's about 1099511620000 characters - masses!) and measures less than a 5.25“ floppy drive and can transfer up to 6Mb per second!!! Only a few years back you'd be buying a 50Mb harddisc for about 600 pounds, so large it needs it's own box and manages less than 1Mb per second. Nowadays you can buy harddiscs that you pull out and take home, or even PCMIA “hard-cards” that look like a fat credit card, or something you'd bung in a SEGA system.

This article is designed to explain the basic working of a harddisc. We shall not complicate the issue with information on SCSI, IDE, ST506, MFM, RLL and the like. Those are all methods of interfacing the harddisc to the computer. Nope. This is about the harddisc.

The harddisc is to floppies what floppies are to cassettes. They offer fast reliable storage that's there when you need it. In fact, it can be argued that harddiscs are the most impressive part of the entire computer system. Sure, the ARM processor is cool - but it doesn't have high-speed moving parts engineered to within microns. You could compare the accuracy to a fully laden jumbo jet flying a metre above the ground at about 500mph – for hours and hours.



The picture at the bottom of the left-hand column shows a very simplified diagram of a harddisc. Each disc (also known as a platter) typically has two surfaces for storing data, and therefore has two read/write heads. The heads move vertically in and out – all moving together – to find the right data. The vertical shaft spins at about 3500rpm, although some drives (such as my Fujitsi) are a fast-spindle type and spin at something like 7500rpm. That means the head is passing over the disc at about 80mph on the outer track. Now do you begin to understand why it is so impressive? :-)

Just like floppies, the data is *tracks*, which is a complete concentric circle of the platter, both sides, and is usually divided into about 32 sectors. You could expect to find upwards of 250 tracks in an centimetre of disc space (~600tpi).

As you move into the centre of the disc, the tracks become smaller. This means the data on the disc is stored closer together, though this is helped by the fact that the heads are not travelling over the disc as fast as they would be on an outer track.

Not all of the disc is used to store data. It is not uncommon to see one side of a disc devoted to the drive itself, for 'housekeeping' and tracking.

The PC world often talks of *cylinders*. A cylinder is all the tracks accessible by a particular head at a particular location. In our diagram, the cylinder would equal ten tracks.

We will digress for a few seconds to introduce SCSI and IDE. These clever mechanisms could give you a disc with 43 tracks and 27 heads or 207 tracks and 1 head (etc etc). This is because the interface no longer addresses the disc by head/track/sector. That is taken care of by the drive. However the drive must 'fake' a certain setup for the computer's benefit. This is less important on modern filing systems like ADFS where things are referred to by zones and SIN (System Internal Numbers) - but look on a PC and your IDE drive could look most odd. Underneath all of the fancy stuff is still a regular harddisc.

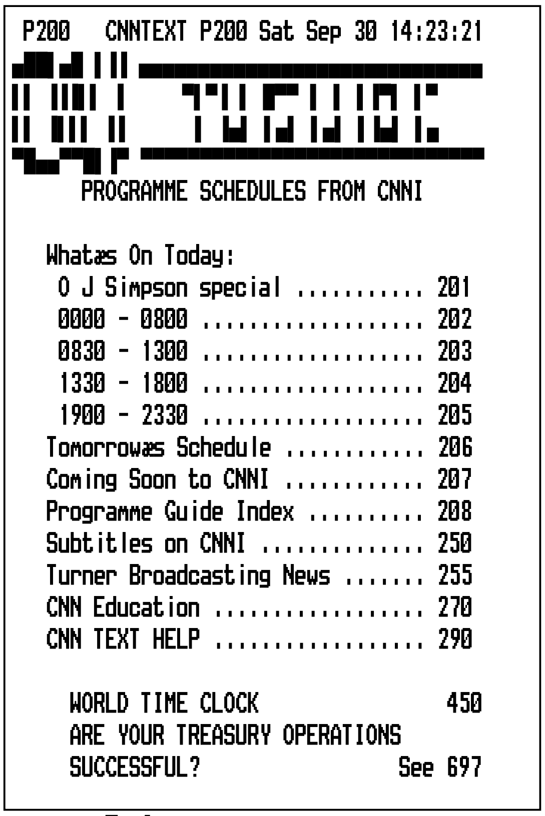
In order to achieve such accuracy, the read/write heads float above the disc at a distance roughly equal to 0.00003 millimetres, held in place by springs and air pressure. It's a distance so small that a mere dust particle could destroy the head. If your drives receive a hard knock, you could crash the heads – literally. I saw a crashed platter. The crash happened roughly in the middle of the platter, looking as if it was car bodywork and somebody gouged into it with a massive key. This spun outwards for about three centimetres and became much lighter. After examination, it appeared that the head had been ripped from the head arm, and the light scratching was the arm. There were large pits and dents all over the various platters where the head had bounced off. There was an awful lot of dust in there as well. So it goes without saying that only a fool would try something inanely stupid like shaking or tapping a working harddisc. Okay, how many people put there hands up? I did. I lost 10Mb of data, from a cheap old drive. Not a big deal, but when you loose, say, 400Mb on your sparkling new drive – it's a major big deal.

This leads me on to my harddisc. My 1Gb. Full of errors. One partition dead. Why? Was I stupid enough to whack it with a hammer? I might as well have been. You see, big modern powerful harddiscs are designed to be switched on. Only going off when the power dies. I wasn't aware of the intricacies of this, so I switched my computer off every night and back on in the morning like I'd always done.

The drive is a fast spindle, and runs at about – oh, 60/70°C – and stays that way. Each night it cooled down. Each morning it warmed up. Those of you that played with crystals in chemistry will know that eventually this can crack a crystal. I doubt my drive is cracked – that would crash the heads – but it doesn't mean that the magnetic coating on the platters hasn't taken on the appearance of crazy paving. I later found out that your warranty is about ten years in a fixed switch-on-and-leave installation, and about 1 year if you switch the thing on and off. Need I say any more? I thought I was saving some electricity by not leaving the machine on over the night. Instead I'm buying a new harddisc for about £350 – hardly a years worth of electricity savings. Need I say more? The internal drives are smaller and cooler and don't suffer this problem. But as any electrician will know – the greatest loading is the surge current at start-up. That's when things are most likely to give up. So, if you can, switch the whole lot on and leave it that way.

Coming soon...

...from BudgieSoft



```
P200   CNNTEXT P200 Sat Sep 30 14:23:21

      PROGRAMME SCHEDULES FROM CNNI

  Whatæs On Today:
   O J Simpson special ........... 201
   0000 - 0800 ................... 202
   0830 - 1300 ................... 203
   1330 - 1800 ................... 204
   1900 - 2330 ................... 205
  Tomorrowæs Schedule ............ 206
  Coming Soon to CNNI ............ 207
  Programme Guide Index .......... 208
  Subtitles on CNNI .............. 250
  Turner Broadcasting News ....... 255
  CNN Education .................. 270
  CNN TEXT HELP .................. 290

    WORLD TIME CLOCK              450
    ARE YOUR TREASURY OPERATIONS
    SUCCESSFUL?               See 697
```

On-line access to TeleText services.

Teletext is copyright. The principle behind this service is users receive teletext free with their TV licence. This door asks users to confirm they have a valid British TV license. Then they may use this door as an extension of their TV set. Useful for people that do not own TeleText TVs...

BudgieSoft will not be liable for situations arising from the use of !TTXdoor program.

Would ***YOU*** like to write for Frobnicate?
Simply write your article and send it to Richard Murray.
Good articles should make the next issue...

KNOW YOUR SYSOPS!

This is my personal insight (and some info) on the SysOps you may meet at the Acorn Show...

DaviD Dade (& Dave Coleman)

One of the most important SysOps at the show is the *guru* himself. David Coleman and DaviD run the Arcade BBS – the premiere Acorn BBS with 5 lines and around 250 callers per day. The BBS is huge, with some 65,000 files. The software is ArcBBS, running on a 13Mb RiscPC600 with nearly 2Gb of storage and an automated tape streamer backup. All lines support at least V.32*bis* (14400bps) with most that I know being able to handle V.FC (28800bps). The top single-person upload count is 1254 files, the top single-person download count is 4146.

Steve Pursey

Steve runs the ArcTic BBS, also in London. He is "not an 18 year old college student 'into computers' but a 40+ year old HGV truck driver...who likes comms". Steve has a wife, three kids and four cats – so he's a busy family man. It's a wonder Steve is with us right now as his first computer was a ZX81, followed by a Vic20. Finally Steve saw sense and bought a BBC micro (:-) go on PC owners, flame me!) and has stayed with Acorns and BBSs since. ArcTic BBS as we know it went on-line on the first of February 1993. By now ArcTic offers two lines, the first with 28800bps capabilities, the latter a 14400bps. ArcTic runs on a RiscPC700 with again nearly 2Gb of storage. I wish Steve and the cats all the best for ArcTic. It's a shame it is so close to Arcade. But only the other hand, if all those Arcade callers buzz Steve on +44 181 903 1309/1308 then... :-)

John Stonier

John Stonier is the SysOp of the up-and-coming Digital Databank BBS, now with 4 lines and and Acorn RiscPC610 and about 1Gb of storage. DigiBank (as it's known in short) offers 14400bps on the three main lines and 28800bps on the main line. John's vision was to "gather as much Acorn information as possible under one roof, and broadcast it to as many people as possible.", thus explaining the name "Digital Databank". DigiBank features a full and useful Acorn section with release notes and updates and such. So when you next visit DigiBank, remember it's aims are:

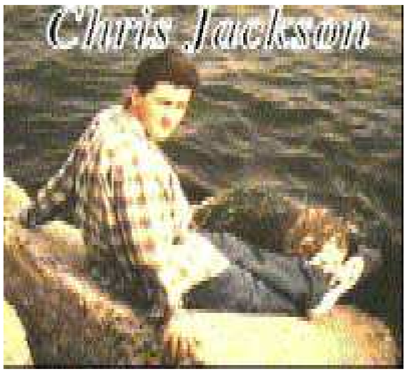
1. To provide a central storage database for all Acorn information.
2. To provide a shopping service for both Acorn businesses and customers.
3. To promote a healthy interest in the Acorn industry.
4. To bring Acorn computers to the attention of non-Acorn owners.
5. To bring like-minded Acorn owners around the UK together for the purpose of stimulation and education.
6. To encourage Acorn owners to purchase comms equipment.
7. To provide the latest demos and information as soon as possible.

Keith Hall

Unfortunately I have been unable to obtain a picture of Keith.

Keith is the creator of the up-and-coming ArmBBS software – the new RiscBBS, as well as one of the people that brought you ArcBinkley. I have been unable to find out much about Keith, but I do know he is a budgie person like myself. :-)

Chris Jackson



On the BBS scene, Chris is probably best known for his Natter SysOp chat door – although he has also written other utilities like ZAnsiDial, CallCost and QwikCD. Chris writes the Acorn User Club Corner and has also reviewed software. The Northern ARM BBS runs on a RiscPC with 1Gb of storage and CD-ROM access. The BBS is also the RiscNet host for the UK and Chris has been most helpful to me in explaining the ins and outs of the ’fido’ system.

Robin Abecasis

Again, no picture. Hmmm... I asked him point blank about a piccy but he declined. He can’t be *that* ugly... I’m pretty d*mned ugly but I released my pictures... :-) Robin is into bikes and leather jackets (so certain people living in northern London should beware). He runs the Renegades BBS up in Scotland (was previously The Wee BBS but Robin hopped on the bandwagon and got ArmBBS and a RiscPC (pah!)). Robin is also known as “Rob the slob” and has written some software under that name – !ArcQuoter being one you might have heard of.

Hugo Fiennes

Hugo is the person behind the ArcBBS software. A lot of work has gone into it. I believe it’s 6 years old. Hugo is a firm comms person with ArcTerm(7) and the SP_Dual serial port also under his belt (ouch!). Hugo runs The World Of Cryton BBS on an 8Mb A540 (finally, no RiscPC!) and about 1Gb of storage. Hugo also has a “very old displayphone” and must be the only person I know that owns one of those things. :-)

Dane Koekoek

(pronounced “cookoo”)

Dane runs his Werewolf BBS on ArmBBS. He has also started WolfNet (competition for RiscNet?) and writes software under the name Werewolf Software. I don’t *quite* know Dane’s fascination with Werewolves, but his ANSI logo is pretty cool... Dane is, for reasons beyond his control, unfortunately planning to attend the Acorn Show on the Friday... Most people are planning to descend on the place on Saturday.

And that leaves only one person.....

Me!!!

See... I told you I wasn’t too hot in the beauty department. :-) I run a BBS called Encina that offers all kinds of bizarre features, including multilingual menus. Due to my not owning a telephone line for the BBS – I hope soon to work out something like 7pm – 7am for the system. However my BBS has no filearea due to the harddisc problems. It’s on magnetic tape...somewhere! I am also known as “BudgieSoft” (or to some, as “BodgySoft” which I think is kinda cute in an extremely sad way). I mainly write BBS doors. Many many of the things. In fact I think at this moment I am the door writer with the most doors (wow! So?). My flagship door is the *CastAVote* suite. A little project that didn’t start out as much, but has become quite something. Other doors of mine are *Parlez, Linker, HappyHak, LastUsers* etc etc. I also write behind-the-scenes utilities for the SysOp: *ReadArea, ViewFile, CloseFile, SetUser0* to name but a few. For door writers there is *DoorDocs*, now in release 5. And for techies there is this creation... *Frobnicate.*
I don’t know if I’ll be at the Acorn Show yet or not. Last year I was until two weeks before when I ended up in Malaga via France! If I do go, I think it will probably be the Saturday. I went to Live’94 on the last day to find most people buggered off home early. So no more last-day-visits for me (though I could have had an Apple Mac for about 500 quid!).

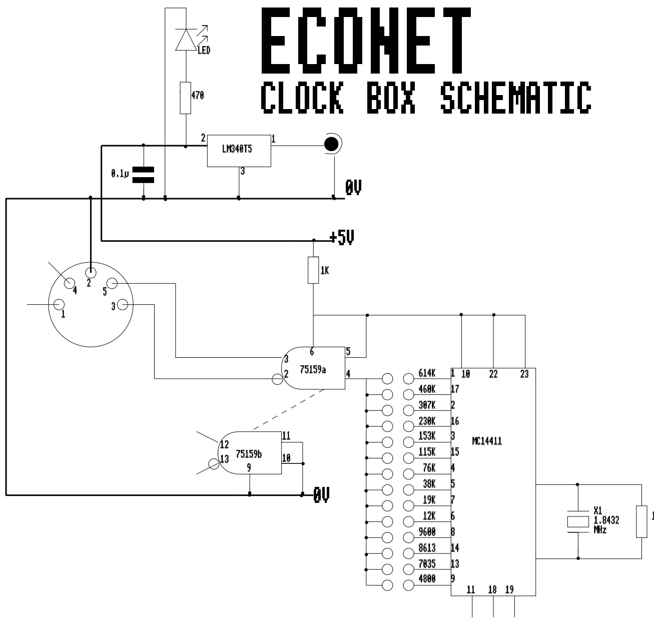
Isn’t it easy to give an insight into yourself, huh?

Oh yeah. My system... Erm... 4Mb A5000 with 1Gb of storage and a tape streamer. AKF12 monitor (and a dead AKF18). Vision digitiser. SCSI card. sp_dual. Two 14400bps modems. BetaMax, VHS and a Canon Hi8 video camera. Satellite receiver and teletext decoder. That’s about it... For now.

For those of you that asked...

Some people have been asking me about Econet hardware. The actual Econet cards are a tad complex to build, so you are best off buying 2nd hand ones. They cost £50 new, so don’t pay more than about £20 for them. You’ll need some good quality 5-core wire for it (shielded telephone cable will work for short distances). You may need terminators. It depends upon your setup.

What you will need is a clockbox. Here is the circuit Acorn details in the Advanced Econet Guide:



The Finishing Touch - addition

Following the last “The Finishing Touch”, several people have pointed out that you *can* tell which Sprites## to use in RiscOS2 – for those that still use RiscOS2, here is some code:

```
spr_len%=&3000
DIM sprite_area% spr_len%
sprite_area%!0=spr_len%
sprite_area%!8=16
SYS “OS_SpriteOp”,256+9,sprite_area%
IF wimp_ver%>=300 THEN
  SYS “Wimp_ReadSysInfo”,2 TO spritesuff%
  spritesuff$=FNzerostr(spritesuff%)
ELSE
  SYS “OS_ReadModeVariable”,-1,4 TO ,,nx%
  SYS “OS_ReadModeVariable”,-1,5 TO ,,ny%
  SYS “OS_ReadModeVariable”,-1,9 TO ,,bpp%
  spritesuff$=STR$(1<<nx%)+STR$(1<<ny%)
  IF bpp%=0 AND spritesuff$=”22” THEN spritesuff$=”23”
ENDIF
IF FNexists(path$+”.Resources.Sprites”+spritesuff$)=1 THEN
  SYS “OS_SpriteOp”,256+10,sprite_area%,path$+”.Resources.S
prites”+spritesuff$
ELSE
  SYS “OS_SpriteOp”,256+10,sprite_area%,path$+”.Resources.S
prites”
ENDIF
```

More about Tornado

Firstly the legal stuff:

INTRODUCTION TO TORNADO:

As anyone who knows me will testify, I don't particularly like RO3.1 or anything of that thereafter. There's a simple reason to this: I think RO2 was the last great update to RISC-OS, and since then there's been nothing added to the OS.

That's why I came up with Tornado. About 80% IMHO of programs written for RISC-OS are written for the desktop environment, and yet the method of using the desktop and writing for it are arcane and haven't changed greatly since RISC-OS 2.

The entire philosophy of writing for Tornado is different. You no longer write applications as programs in their own right. You write code which defines your program as what you want it to be. You don't write code to redraw windows, you don't write code to open menus, you don't write code to load or save files. You don't write code to manage memory. You don't write code to recognise the difference a double click from a triple click. You don't even write code to load in your user-defined configuration.

The whole thing about Tornado that will scare most programmers is how little you *do* do. And the great advantage of that is that the Tornado operating system suddenly has a lot of power over every app using it - and the great advantage of this is it allows applications to function as a single unit in a fashion never before seen on ANY platform.

And despite all this automation, Tornado increases the power available to a task by incredible amounts. Suddenly, things like printing no longer take over the desktop. Tornado applications can just as easily print five files, while recalculating a spreadsheet and converting a set of files from GIF to JPEG ***on a 1Mb machine*** (it uses virtual memory) as allow you to move the mouse around the screen. And this BTW, doesn't require any applications loaded in other than a single spreadsheet.

Tornado offers processing occurring on a task at multiple levels, via a mixture of multithreading multiprocessing and its TAOS-like subtask facilities. Files loaded into one application can appear in another application's window without consuming any extra memory, and any operations performed on one 'view' affect all other 'views', as actually they are all multiple views of the same file [1]. Files can be OLEd and RAM transferred to and from any app, including non-Tornado one's (although OLEing between Tornado apps is _much_ more pleasant and convenient).

The foremost objective of Tornado *is to increase productivity.* Above all else, it will do this to its fullest extent. The second objective is to be frugal with resources ie; be quick & responsive, not consume vast amounts of memory to do simple operations (like certain PC GUIs), and not to take up vast stretches of disc space. The tertiary objective is to remove as much programming time from the programmer as possible, and to make life on him/her as easy as possible.

[1] It would seem some people don't quite understand this. A view of a file is updated in all views if any of the views are altered (this might seem like it's slow, but remember that while all the displays are updated, you can work on as tornado multitasks during window redraws, unlike RISC-OS). Note also that a view can be marked as a separate file - and should the view marked so be altered, it is made into its own copy, delinked from the other views and then altered. This way, memory isn't wasted on multiple copies of the same file.

I'll also mention here that loading a file into one app, and then the same file into another will make tornado actually consider both files to be the same (and thus one a view of the other) - using only one copy of the file - but should either be changed, they are then delinked and two copies appear in memory.

Frobnicate detailled Tornado, in Niall's own words, in issue 2. However the Tornado project has moved on from a pre-emptor to a whole operating system.

We would, however, like to point out that programming something of this complexity is a long job. Frobnicate will try to keep you *in the picture* without the usual hype. Also, on behalf of both Frobnicate and Niall - please don't email Niall unless you genuinly have something to contribute. Information on Tornado will appear here and on DigiBank - as well as other ftp sites. Let Niall get on with this, and we'll all see results sooner.

Part 1
What is Tornado?

Many of you reading this will already know of the movement in the Acorn world – a sort of revolution if you will – involving tornado. And while much misinformation abounds, there are a few things which are fact:

• Tornado has been designed and is being written by Acorn users for no direct monetary reward whatsoever. It is being produced by writers entirely independent from Acorn, because they are tired of Acorn's reselling of the same product and unwillingness to make changes.

• Tornado does not come on a set of ROMs like RISC-OS. It actually comes in a set of resources contained in a directory. When this directory is seen by the filer, tornado is loaded into memory and the tornado environment set up. From hence tornado applications can be executed under the tornado multitasker, a piece of engineering far more sophisticated than the RISC-OS Wimp Manager. This multitasks code preemptively, and allows a level of multitasking far beyond that seen before on Acorn machines.

Tornado is split into (currently) four parts. There is the kernel, which provides various extensions to the RISC-OS kernel not currently available eg; better memory management, and mouse control - this part of tornado can be used by any code in the system. Next is the tornado shell. This runs under the RISC-OS Window manager as a wimp task, and under that then manages the various tasks running on tornado. Next there is the tornado hacks module, which isn't really part of tornado but as it needs both the tornado kernel and shell it is packaged with them. It hacks into the existing window manager, and provides things like multitasking disc I/O (including loads and saves). Finally there is the tornado filing system, which among other things interfaces between tornado and the RISC-OS filing systems – the main difference being that under tornado you can have unlimited length filenames. TFS does this by creating a single file in the root of a filing system, and then manipulates the storage of files in that single file, like an archive. This bypasses the existing restrictions of filecore. Also, TFS provides a RMA based filing system, tfs: which is used by TShell to cache and optimise various operations. THacks also requires this module to operate correctly.

Tornado is very future proofed. It uses 32 bits to describe colours used, and uses 32 bit filetypes too internally. It is not reliant on any particular architecture, and can utilise up to 4Gb of RAM. It also can load files up to 4Gb into tornado applications using only as little as 512k. Almost every part of tornado can be intercepted, or hacked into, or redirected. This allows infinite possibilities of future expansion into the next century.

Finally, tornado's first and primary aim is increase productivity. This means, conversely, that anything hindering productivity will be axed, and 'gadgets' designed to impress the ignorant masses will not be tolerated.
Tornado is a powerful, functional operating system – it will not mollycoddle newbies, it will not pull punches. It will not use simple language to users. Users will be expected to know what they're doing, and if they don't then they should find out.

This does not mean there will not be an extensive help system available. Complex hypertext documents and interactive help can be bundled with applications, in order to lessen the learning curve (which when high impedes productivity!), but they can be deinstalled (thus saving disc space and processor load) when no longer required.

Well, that's a summary of tornado. The rest of this information goes on to explain tornado in greater detail, and there are docs available which will even break down the internal data formats and protocols used. They are not generally available, as few can really understand them, but every document will be available from hensa and the Digibank BBS - ie; don't email me and ask 'cos they're available from both those sites!

Part 2
Memory management

Tornado manages its memory, and the memory of the apps under its control much differently from convention. The application image is stored in its task slot, allocated from the tornado memory pool, along with any internal data. Thankfully, unlike RISC-OS's 16Mb wimpslot limit, tornado applications enjoy a 4Gb maximum slot size. This part of the memory management is handled by the tornado shell.

All files loaded into tornado apps are stored in a heap located in the tornado multitasker's memory space (on MEMC1-1a machines the memory space is a standard wimpslot [1]; on later hardware it moves into a dynamic area). This heap's blocks are relocatable, and the heap is garbaged regularly. Any free space is returned immediately to the wimp's free pool. This part is handled by the tornado kernel.

[1] Note that on current implementations of tornado, the public area space (tornado system heap) resides in RMA, due to it being a lot easier to debug things when your heap doesn't keep getting corrupted. It is envisaged that in the future, files loaded into tornado will be kept in the main memory pool, and virtual memory applied there. However, as far as the application will be concerned the files will still be stored in the tornado system heap.

The reason why files are kept in tornado's memory space is because:

(a): On every architecture, this method allows a slot up to machine RAM size. Obviously, this allows files of almost infinite length to be loaded when possible.

(b): Virtual memory techniques can be applied to data, whereby blocks (being relocatable) can be dumped to disc and copied back in when needed (i)

(c): The files are at the full reach of Tornado (it owning the memory), and so can be accessed without the owning task's permission. Obviously tornado apps must keep a copy of the file in a saveable format at all times - if this is not possible, the task must provide a routine to generate a saveable file.

[1]. Of course, no task actually owns any files loaded into tornado apps - except Tornado itself.

Advantages of this system of centralising files:

• Saves and insertion are/can be automated

• OLE and hotlinking can also be automated

• Converters can be used to translate between formats. For example,

if the user drags a GIF file to a Tornado app that can only accept Sprites, it gets converted first and dumped into tfs: to be loaded in. Being saved back, it get converted back. This is done by a specialised range of subtasks (see the appropriate docs)

• Upon your app crashing, files currently being edited can automatically be saved out. This of course can only be done if your app maintains a ready-to-save copy of the file at all times (which is recommended). (See appropriate docs)

• You can use the Tornado renderers. These centralised renderers take a file, in its saved format, and render it. They can render 2d vectors, text, DTP pages, sound, 3d vectors, bitmaps, soundtrackers, movies – whatever. Once a renderer is written and installed, it's available for all apps that can take it (see appropriate docs)

• You can use the Tornado multiprocessors. Although these are far more generalised than just for this one, these can convert between file formats (as mentioned above), but can also do complex tasks eg; rotate a JPEG. This may be done not only on a local processor, but perhaps on a second processor or a processor on a machine on the other end of a network (see appropriate docs)

(i): Virtual memory works by breaking up very long block into much smaller ones (usually a multiple of the track size of the disc media upon which the cache is being kept for speed). Each of these blocks has a time associated with it (OS_Monotonic) ie; when it was last accessed using Tornado_Getaddr. When necessary, blocks past a certain age (or type - for example, blocks all belonging to a 25Mb block will be swapped in and out depending on which part of the total block is being accessed - thus not flushing all data onto disc) get dumped to disc and it is stored where they are (disc pathname). This allows a 2560x2048 32 bit sprite (taking normally 20Mb) to be edited on a 1Mb machine, as given that there would be an average 75 byte header, and with 2048 blocks of 10240 bytes each (=2 tracks on floppy E format) = about 170k of memory + 10240 to store each block.

The common tornado heap help in RMA is a special tornado heap, and is referenced by passing 0 as the heap start to the Tornado heap SWIs. It features relocatable blocks with auto-garbaging, and fixed blocks are supported too. Heaps are also relocatable, so you can have a heap in a heap. Also, heaps can be auto-extending, ie; they will increase/decrease the allocation of whatever they are stored in. In the case of postslot heaps, they will automatically call Tornado_ExtendSlot to alter the size of the heap. This makes setting up postslot heaps extremely easy.

Blocks in these special heaps are referenced to by their handle, a negative number. This handle is passed to all heap operations, and the program need not worry where the data is really stored. When it needs to access the data, it calls Tornado_Getaddr, which returns the current address of that block. Fixed blocks are referenced a la OS_Heap style, with the handle being a fixed address in memory.

Another heap used by Tornado is the subtask heap, referenced by 1 – but this is for the exclusive use of subtasks – see the appropriate documentation.

Part 3 The Tornado shell

Well, this time I can actually write something about the tornado shell - some it has been written, and much more finalised.

When you start up your RISC-OS computer, and display a directory containing !Tornado (tornado's resource directory), it loads itself in and installs its resources. TShell sets itself up as a Wimp task, and claims a section of application space on MEMC1/1a machines, or a dynamic area on later architectures. It then starts up the multitasker in the memory space, and starts up a small task which handles tornado's user interactions.

From hence, all files double clicked on in the filer are intercepted, and checked to see if they are wimp apps. If so, they are returned to the wimp, and if not a new memory domain is created, the image loaded in, and execution started.

The code is now preempted, and a handler is installed on WrchV which upon vdu output creates a 640x512x2 sprite, and redirects vdu output and the OS_ReadVduVariables into it (thus allowing most games to work). The task can run nicely in its window, or if the user chooses they can move the task to full screen use. Note that during full screen use, either full, partial or no multitasking is retained [1].

On the other hand, if the image calls Tornado_Initialise before outputting to the vdu, the handler is removed and tornado searches the home path for the options, templates, messages and menu files. These are all loaded in by tornado, and installed in memory.

Writing for tornado

Code written for tornado is done using a visual editor, similar to that used on other platforms. You create an application, and then design the windows to be used in it. To the various parts of each window you attach code segments, written using Zap via the external edit protocol. Probably the best way to explain this is to write a program, here and now. Ok then: how about a program which will update a window in real-time according to the amount of space available on a disc drive?

Firstly, you create a new application (which appears as a icon in the loaded apps window), and then you create a window in it by dragging the ‘create window’ tool onto the window icon. The new window pops up. Inside the window you create a text icon with “Free:” in it by dragging one of the predefined text icons into it, and beside it a green raised icon - also from the predefined icon library. Now, you use the BASIC link tool to attach a section of BASIC to that icon, and make the code callable on every refresh. Up pops an empty function in a Zap window:

```
DEF FNicon1
  LOCAL

=
```

Now, you change this to:

```
DEF FNicon1(disc$)
  LOCAL free%
  REM By right I should check for other FS's
  SYS "ADFS_FreeSpace",disc$ TO free%
=free%/4096
```

... and change the code entry to ‘wi=FNicon1(MID$(”%ti”,1,INSTR(”%ti”,”:”)))’. This, by the way, set the width of the icon this code is attached to to the returned result of FNicon1. FNicon1 is passed the result of ‘MID$(”%ti”,1,INSTR(”%ti”,”:”))’, where %ti is replaced with the text in the title bar (which BTW would have code attached to it setting it to the media this window is referring to). From hence, every time the window is opened this code is called, and the bar set to the correct value. But this isn’t real time, is it?

The trick is to set up a ticker. This is done by specifying a ticker countdown in cs, and every x cs the code is called. Here’s the code:

```
DEF FNticker1(handle%)
  IF !updated% THEN SYS "Tornado_OpenWindow",ha
ndle%
=0
```

This checks !updated, and if it’s true reopens the window. BTW, the link code for this would be: ‘FNticker1(%th)’, where the handle of the window owning the icon is substituted in for %th.

But how would !updated% be altered. By setting up a handler, in this case a window create handler. This code gets called on every window created:

```
DEF FNhandler1(wkspace%,disc$)
  LOCAL P%,N%,blk%,addr%
  SYS "Tornado_Getblk",%1,0,1024 TO ,,,blk%
  SYS "Tornado_Extblk",,0,+4,wkspace%
  SYS "Tornado_Getaddr",,0,,wkspace% TO ,,addr%
  !addr%=blk%
  FOR N%=0 TO 2 STEP 2
    P%=blk%
    [OPT N%
    .disc EQUS disc$:EQUB0:ALIGN
    .upcallv
    \Checks for alterations to the path at .dis
c, and sets .updated% if true
    ...
    .updated% EQUD 0
    ]
  NEXT
=1:REM This specifies we want to keep the wkspa
ce passed to us
```

Of course, this will only allow one window to work at a time. To get around that, you would put updated% in the wkspace. The wkspace ptr can be passed via a substitution to any code linked to icons, handlers or tickers called from that window.

Of course, future versions of tornado will allow the ticker used above to be done automatically by tornado - tornado will check updated% for the task. But for the time being, this is as far as tornado goes.

Languages

Although I used BASIC to demonstrate tornado’s visual style of writing, both C and assembler can be used as well. In fact, there is no reason why a mixture of C, BASIC & assembler can’t be used in the one program, and leaving the tornado editor with the problem of linking the lot together. Also supplied with the editor are libraries of code segments, which allow further standardisation and ease of program construction.

However, despite the loss of multithreading with BASIC, BASIC does contain one huge advantage over C and assembler - it’s interpreted. This allows tornado to directly call functions in the image using EVAL. With C and assembler, large tables of function names and their offsets must be created – a real burden for a programmer. Of course, the editor will automate all that – but for now, it’s a lot easier and simpler for us to write the lot for BASIC - and extend for C and assembler later.

Anyway, BASIC needs a bit of a boost. Acorn have definitely been ignoring it in the last few years.

Part 4
Crash protection

From the moment a program starts up under tornado, it comes under the protective wing of Tornado. From now on, there are very few ways programs can lock up the machine [1].

Tornado monitors all filing system operations done by the app/program, and closes any files that are still open when an app/program terminates (unless vetoed by a service call or wimp service message broadcast). Also, it installs handlers to deal with Undefined instructions, Prefetch aborts, Data aborts, Address exceptions, Branch through zeros, CAO exiting errors, normal errors and when OS_Exit is called.

For Undefined instructions, Prefetch aborts, Data aborts, Address exceptions, Branch through zeros and CAO exiting errors, preemption is halted on that task, control removed and a window is displayed telling the user that this fatal error has occurred, and asking what should be done about it. The user then has the option to save out any files currently loaded into that app, or to save them into a temporary space and restart the app, which will then reload in those files, or to ignore the error and continue (in which case you'll get the usual error box, and lose your files). For normal errors, usually the task has its own error handler installed to deal with these errors, but it may request that this operation be automated.

For OS_Exit being called before Tornado_Closedown, tornado cleans up, again by asking the user about any unsaved files etc. etc.

There is also another level of protection: If the task goes into a never-ending loop, and if the messages waiting for it exceed a certain value, a message pops up to the user indicating that it is most likely that the task has crashed, and does the user wish to terminate the task.

This leaves only a few ways left of locking up the machine, ie; the ones which would usually cause a full reset to get out of them. Due to the structure of the RISC-OS kernel, it's very difficult to get around these, but I'm sure you'll agree that the proposals above with certainly help no end.

Part 5
Subtasks

Subtasks are an extremely powerful way of doing processing. They are TAOS-like in the way they can process in multiple threads. The benefits of TAOS are well known, and most of those benefits are available to tornado apps.

A tornado subtask is a normal tornado task, except that it is subject to a number of restrictions – they are only allowed to access memory within their own space and within the subtask heap, and can only communicate normally with their parent and their sibling subtasks. Generally speaking, they fill a buffer of processed data and pass this to their parent ie; they do their processing in sections.

The main advantage of using subtasks is that it allows a bit more of multithreading than usual with BASIC. Due to BASIC's structure, execution cannot occur at different parts of the program at the same time, so by using subtasks a BASIC programmer can still have limited multithreading.

Another advantage is that with the arrival of multiprocessor Acorn's, the subtask can be executed on a separate processor - and thus giving all the obvious advantages.

An example of this is a http fetcher. The parent html viewer, is asked to fetch *http://www.acorn.co.uk* for example. The parent, a tornado task, starts up one of its private subtasks stored within its directory (some subtasks are stored for public use ie; any tornado task can use them) which fetches http: links.

The subtask starts, and uses ttcpip: to send a http request. When receiving confirmation, it passes the message to its parent which displays the appropriate message. Then the html page arrives in packets, as is in tcpip's nature, and this is run through, picking out all the references to graphics/sound/movie. The graphics/movie are then sent for, by the subtask starting up siblings, which in fact are copies of itself. Each of these siblings reads in the graphic/first fram of the movie, checks what format it is, and then starts up the appropriate graphic converter from the public converter library, passing it the reference by which it can send its processed data to the parent app.

Ok, so now the picture looks like this:

```
                        Parent html (www) viewer
                                   | Passes down http://www.acorn.co.uk
                                   |
                                   | Passes up html document
                              http fetcher
 Passes down http://www.          / | \         Passes down http://www.acorn.
acorn.co.uk/     ---------------  |  ----------------   co.uk/sprite.gif
drawfile.gif   /     Passes down http://www.acorn.     \
               |            co.uk/movie.rply             |
               |                   |                     |
               |                   |                     |
        http fetcher        http fetcher           http fetcher
 This being a private        Passes | down all     Passes | down all the
subtask knows its parent  the data | received       data | received
can plot drawfile, so it           |                     |
sends its received data            |                     |
straight back to its      Passes up | when to stop    GIF=>Sprite
parent. If it were a        sending | data             converter
public subtask it would     Replay=>Sprite         This takes the GIF
check what filetypes the      converter            and converts it into
parent can load, and       This takes enough       a sprite, and sends the
convert to sprite if it    data to convert the     data, as processed, back
couldn't handle drawfiles  first frame into a          to the parent
                           sprite and also sends        / | \
                           all the data processed  Delegated tasks
                           straight back to the    The same as below, except
                                 parent            obviously it's multiple
                                 / | \             copies of the GIF=>Sprite
                            Delegated tasks           converter
                         These are separate copies
                         of the Replay=>Sprite
                         converter called when the
                         converter above this is still
                         processing but is asked to
                         process another segment of
                         data retrieved from ttcpip:
```

As you can see, processing of all data coming in performed simultaneously, which means that no matter how quickly the data comes in, it is all
processed, not held up until the processing and conversion routines can deal
with it. The speed and productivity gains are impossible to calculate. It

also means that the parent html viewer may have bits of GIF appearing in a seemingly random order, which I would suggest is not altogether bad.

Subtasks are also subject to constant monitoring - should one fail due to lack of memory, it's parent is notified and can do what it likes with the information eg; schedule a retry, or perhaps inform its parent (if a subtask). It may also display a message/put a question to the user via Tornado_Query.

Should a subtask crash, it gets the same treatment as other tornado tasks – except that again its caller is notified after everything has been cleaned up. Subtasks can also request that a postslot heap be set up, so they can store internal memory allocations.

Another feature of subtasks is that all vdu output is sent to its parent. This can be used to run normally single-tasking tasks eg; my hopefully forthcoming animals guessing game door for Newsflash will be the raw program in BASIC as written in 1993, but with a parent 'supervisor' which starts up the program as a subtask, reads in all graphics input, converts any vdu colour sequences/cursor positioning into suitable ANSI commands, and spits that down the line to Newsflash.

Special subtasks

Special subtasks currently come in as file converters. If a user drags a GIF file onto a tornado app which can only accept Sprites, then the converter subtask (stored in tconvert:) list entry called by the 32bit hexadecimal of the GIF filetype is checked out to find out what that filetype can be converted to. If it can be converted to a format that is loadable by the app the file is going to, the GIF file is loaded into tfs: while multitasking, and the subtask, as specified by the subtask list, is started up. The parent, Tornado, sets up an area of memory to receive the file, and the subtask starts spitting out converted Sprite.

When the subtask finishes, the original GIF data is thrown away and the file is loaded into the app by Tornado (see the appropriate document about loading and saving). Voila!

On saving, in the save box a GIF filesprite is shown, and unless the user changes it, the file is saved to tfs:, converted back by the appropriate subtask (this time determined from the entry for the hexadecimal of the sprite filetype).

Other things about subtasks

Another thing to be noted here is that Tornado is intelligent about subtasks. It caches the most recently loaded subtasks so that they may be accessed quicker, if the user so wants, in tfs: (if present).

As said above, subtasks may not be necessarily executed on the local processor. They may also be executed on a second processor, or on a processor running somewhere on a network. Many people laughed at my original claim that the processor may be running on the other side of the world, connected via Internet. This is not unrealistic. Because subtasks communicate by Tornado_SendSTMessages and Tornado_GetSTMessages, and think they are the only process running in the machine, they can be run by a suitable multitasker/server. A client running on the local machine can send the subtask code down the network to the server, wherein it is cached and can be kept for future use. All messages between parent and subtask can be run over the network. Easy!

Summary of subtasks: they are extremely powerful, and their importance in the Acorn world can only increase, if not within Tornado but by the probable implementation of TAOS for the ARM series of processors.

Part 6
File renderers

Generally speaking, all files can be displayed by Tornado apps, whether they be text, styled laid-out text, bitmap, vector, 3d vector, sound sample, soundtracker, movie etc. This is done by the following method.

A file is loaded into a Tornado app which can load such filetypes by Tornado, and is asked to redraw the file. The app calls Tornado_Renderfile, with the filetype. Here's what happens at this stage:

- Tornado broadcasts a service call, and if a module picks it up, it renders the file.
- If the service call isn't acknowledged, Tornado returns 'Please reschedule' to the calling app, gives the user a message saying 'Please wait', and looks in TRenderers:List, at the entry which is the 32 bit hexadecimal of the filetype being rendered. The module detailed in there is loaded in under multitasking.
- The module is initialised, and loads in any libraries it needs under multitasking. Tornado first checks to see if these modules are present - if so, all well and good, but if not they are loaded in under multitasking. Tornado makes a note of the modules it loads in.
- Meanwhile, the initiating task has been calling Tornado_Renderfile repeatedly and being told to wait. Now the renderer is present, the service call is rebroadcast, whereupon the module should render the file.
- Now, depending on what the user has configured, the renderer is told to quit and all modules loaded in by it are also quit, and removed from memory.

Obviously, if the user wishes, the module may remain in memory (the user specifies max renderers to be in memory at once, or max number of bytes they and their libraries should take up etc).

The reason for the libraries is that most files are currently rendered by module-based code eg; soundtrackers, artwork files etc. All it would take is a simple suitable module front-end which interfaces between Tornado and the appropriate library.

As you can see, this is another extremely powerful aspect of

Tornado. It means things like DTP and hypertext apps can have all types of file integrated into them, only limited by the renderers available. This includes movies, sound, graphics – whatever.

Part 7
I/O, OLE and hotlinking

Tornado does all disc I/O while multitasking, including when an executable is being loaded in from a !Run file. Tornado does this by providing *-commands to load in code while multitasking. A background operation hourglass appears during this. Also, any I/O operations are done by Tornado, including serial, parallel, disc and inter-application I/O. Sometimes the I/O is done without the application's knowledge or involvement. For a start, all low-level serial and parallel I/O is done using SWI Tornado_IOOp. Block gbpb taking more than 1cs to perform gets broken into blocks and transferred while multitasking, although the calling task can have the SWI return control instead so full application functionality is retained (BASIC programmers would need this the most - C and assembler images can be multithreaded in this case). I/O is done using the serial block drivers for the serial port, and with a specialised replaceable driver for the parallel port.

For more generalised use, I/O can be directed at tserial:, tparallel:, tprinter: and anything else that is added by third-party producers (eg; tethernet:). For the printer, tprinter: in fact is a FIFO buffer which takes in input and spits out output to the printer as fast as it can. I/O to tprinter: is also done while multitasking, unless the app wishes to retain full functionality. I/O to tserial: actually directs I/O to whatever is the currently selected serial driver(s) (can be different for different apps). Also, multiple apps can access the serial port at once, using a system like the input focus. The access requirements for eg; tethernet: have yet to be finalised.

That's general I/O. For more usual I/O like disc I/O, there's a different setup. For a start, applications do not know that a user has requested that a file be loaded into that application. Nor does an application know if the user saves the file out of it. Or if the user has OLEd a file currently loaded in. Or, for that matter, if the application has any files loaded in at all. [1]

How is this possible? Well, tornado applications are written to edit one file and one file only. **Tornado** handles the loading in of multiple files. Also, tornado applications do not *load* in files, they rather replace an existing (possibly empty) one. For example, loading in a tornado app and then loading a file into that actually replaces the blank file which is considered to be already loaded in with the one being loaded in. [2]

This means that the user OLEing a drawfile in a DTP frame simply does the following:

- User does OLE special-key & click.
- Tornado receives this, and checks the filetype of the file loaded into the frame in the DTP (i)
- Tornado checks the apps currently loaded in, and sees if any of them can load the file (ii), and if none are loaded in it checks to see if any suitable apps have been seen by the filer (iii)
- Tornado tells the receiving app to *replace* the currently loaded blank file with this file here, and passes it the address of the file loaded into the DTP package. It also tells the new-found editor where to open its editor window if appropriate eg; directly over the frame containing the original, scaled to the correct zoom factors etc. (iv)
- From now on, any modifications made to the shared file is accompanied with a message sent to all tasks with access to that file requesting that they redraw that file (v)

(i) Tornado knows the filetype of every file loaded in because *it* does the loading in.
(ii) Tornado knows which filetypes each app loaded in can edit (without conversion) because the app declares them in its Tornado script file.
(iii) Tornado knows which filetypes each app on disc can edit because they are declared in the !Boot file of every app on disc. Tornado can build a 'map' of the disc by checking the disc for tornado apps, extracting what files they can load and compiling them into a list. When the Tornado filer is written, any changes made to the directory structure (ie; the user moves a file from here to there) will also update the list. Otherwise, or to save memory, Tornado can simply register any apps seen by the filer.
(iv) All files loaded into all tornado apps are stored in a central reservoir maintained by Tornado, and this lives in RMA. See the appropriate file about this.
(v) All files are redrawn by the Tornado renderers, which are available to all tornado apps. See the appropriate file about this.

From the above, you can see how the user dragging a file from the filer to a tornado app would work, and dragging a file from another app to an app, and hotlinking a whole bunch of files together. It's even possible that a single file could be shared over a network eg; an alteration to a file on one station would affect all copies all over the network (and beyond). This would allow teachers to monitor what any student is doing.

Note also, that all files loaded in and saved out go through tfs: (if present). The file is loaded while multitasking into tfs:, and loaded into the app from there. A file is saved to tfs:, from where it is copied to disc under multitasking. This ensures constant multitasking.

See also the section about subtasks, as subtasks play a more than notable part in implementing I/O.

[1] Note that in fact, no tornado application can have a file loaded into it. The file in fact is loaded into tornado, and tornado passes the editing of the file to the app the user assigned to it by dragging the file onto the icon of that app. Ie; the app doesn't maintain its own list of files. Nor can it distinguish between files passed to it by tornado. When a user alters a file, tornado informs the app owning the editor window and passes it the address of the file. The editor modifies the file, and tornado shows any changes onscreen.

[2] In other words, multiple files 'loaded' into an app are really a list of files associated with that app. When one is altered, the app is called to do it. As far as the app is concerned, it can edit one file and one file only – whatever the file tornado passes to it.

Part 8
What tornado will *not* do for your machine

While tornado is brilliant and all that, there are just some things it won't and cannot do. These are some of them:

Tornado will not speed up your machine. Anything but. In fact, a RO2 Arm2 machine with tornado running is just about usuable in a hires mode (but remember a RO3 Arm2 in SVGA is so slow it's almost useless). Tornado's advanced features do not come without a price, and both memory and speed suffer under tornado. While tornado works fine on a RO2 Arm2 Mode 12 machine, it's pretty much useless when you're using mode 21.

Also, if you're planning to do any demanding computer use with tornado, don't bother with less than 4Mb of RAM and a HD with 100Mb free. It'll work quite nicely on 1Mb, but don't expect to have more than two or three applications active. Also, your hard disc will sorta go spastic if you're not careful as 1Mb fills up very quickly indeed, and then the VM kicks in. 2Mb should be fine for typical use, but I'll put it this way - I wouldn't want to write tornado apps on a 2Mb machine.

Simplistically, tornado will not make a RO3 machine go as fast as a RO2 one. Simply can't be done I'm afraid. RO2 had some parts hardwired - which is why it goes so fast.

Tornado will not make BASIC multithread. Quite simply, the structure of the existing BBC BASIC won't allow different parts of the executable to be running at once - or at least not without a lot of memory waste. With care, C and assembler can multithread quite nicely, but you do have to be careful when writing them.

However, subtasks can still be written and called from in BASIC. These allow a cumbersome but effective method of multithreading processes.

Tornado does not provide full virtual memory. It only provides virtual memory on memory blocks held in its system heap - and this allows files of almost unlimited length to be loaded in and edited. It will not allow private areas of memory to have virtual memory performed on them, nor will it allow code to run in virtual space. The structure of the RISC-OS kernel prevents virtual memory working correctly when applied to code images.

Also, may I add that even if it were possible, I wouldn't allow it. I have many objections to full VM, and personally only see it as a good method of editing files larger than memory. No more.

No doubt, this list will grow. Keep watching this space ...

Finally,
Part 9
Miscellaneous

Well, this is simply here, as the name denotes, to stick in all the things not covered by the other parts of this archive. Or, in other words, to boast about Tornado!

Tornado is heavily future-proofed. It has provision for infinite length filenames, infinite files per directory, and interestingly 32 bit filetypes. Internally it uses all of these, and 'translates' to and from what is currently used when required. It can handle multiple processor architectures as and when they appear. It's extremely flexible. Tornado will bend to even the most demanding applications, and can be customised to a level pretty much unheard of on any architecture. Routines can be replaced by 'fix-it-up' modules which correct bugs.

Tornado currently is written entirely in assembler, and it is envisaged it will remain that way. Tornado is fast, frugal and sucks as much power from the processor as it can. It should only use about 200k of module space (I'll remind you about all the things it does!).

All of Tornado's code is reusable eg; a piece of code it might use to put a routine on the SWI vector will always have a SWI attached to it, so that other programmers may use Tornado's code. Tornado's code is written above that of the current norm, with extremely flexible abilities, proper error handling and also, it's fast.

Tornado supports multiple users on one computer, and stores config files in such a way that they are different for each user.

Also, little desktop niceties will be implemented eg; a hotkey which cycles through the currently open windows (including RISC-OS ones), bringing each to the front and giving it the caret. And a hotkey which opens the directory which the pointer (which is dragging a file into the filer) is over – thus stopping the really annoying times when you drag a file out to find the directory you want to save into isn't open. And a hotkey & menu option to send/take all selected stuff to a certain app each time. This implements an extremely effective intertask clipboard, and the file will be converted if necessary.

Finally, some may have heard that it is intended that the RISC-OS desktop be rewritten to make best use of Tornado's facilities. This is a long way off, but it may become impractical to have the filing system still running on a RISC-OS level and everything else running at a Tornado level. Rewrites of the filer, display selector, and definitely task manager will certainly be on the cards. I have a load of things that will be done if this ever happens, thanks to the people from c.s.a.*. But for the moment, it's not happening.

Other miscellaneous things

Also some may have heard of the Tornado verification procedure. Essentially, commercial writers writing code using Tornado will be encouraged to send in their program for testing, for a fee of course. The program will be meticulously tested, any problems (if any) noted and the writer informed as to whether the program has passed or not. From hence, the writer may quote in all adverts that the task passed the test, that it conforms to certain basic guidelines. Thus, a program with the test passed with have a considerable advantage in the marketplace over a non-passed program, as the consumer will know that a program which has passed the test will be guaranteed to have certain 'niceties' which other programs may not. Also, copies will be kept of the report sent back to the writer, and made available for public inspection so that potential buyers can check out a program before buying.

Far away things

If Tornado really takes off, and for that it needs to be written first!, it can be expected that a professional commercial suite of software will be released which will be based around the VisualXXX available for Windows, but definitely without the crippling disadvantages that those programs have. Essentially, it will allow tornado apps to be written very quickly, and that will justify its price tag. Users will still be able to write tornado apps without it using the shareware editor, but obviously it will be slower and more niggly.

Up until this commercial release, that limited shareware visual editor will also be made available for use by writers not wishing to fork out for the full development suite. This shareware editor will be maintained and upgraded on an indefinite basis – however, not to the extent that it would trample on features found in the commercial version. :-)

Other ideas include building in software encryption/compression in on all I/O streams. In other words, as all I/O is done by tornado, it will be very easy to compress/decompress data on the fly.

Another thing we'll get around to eventually is implementing long filenames and infinite files per directory. This will be done in conjunction with the Unix zip suite of programs & TFS, and will work by creating a single archive in the filecore root directory and then doing all I/O to and from this archive. Since data can be quickly compressed to 50% almost on the fly for a floppy disc, it's not out of view that the archive will be a full zipped compressed archive.

And finally...

Finally, tornado will remain public domain for all time, and will be made as accessible to users as possible. Development on tornado will go with what the users want, not what the writers *think* they want. Unlike Acorn, we want to make a difference, a change – rather than constantly refining the same RISC-OS 2 over and over again, which is ultimately self-defeating. Grumbles in the Acorn market are growing stronger every day, as acorn users watch other platforms GUI's get better and better, and yet the RISC-OS GUI stays pretty much exactly where it is. Increasingly, MCIBTYC arguments are becoming harder and harder to win. Which, for any Acorn-loving fanatic, is enough reason for violent action! :-)

Cheers, Niall,

at ndouglas@digibank.demon.co.uk
at Niall Douglas@Fidonet#2:257/501.13 or Riscnet#7:353/1.0

Remember, don't write Niall messages unless they are *useful* messages. Specifications are kept on Digital Databank BBS and some stuff will also be printed up here. To add to that, Niall lives in Eire, has to pay to collect mail – and any self–respecting comms person knows Telecom Eirean (sp?) isn't exactly cheap...

DigiBank can be called on:
1 line, 28800bps +44 1707 329306 23hrs
3 lines, 14400bps +44 1707 323531 24hrs

In a future issue, Frobnicate may print some of the proposed protocols and other specifications. But first, Frobnicate would like to offer this for your delectation:

Cliff Dobbs, cdobbs@armltd.co.uk

Oooo, a person from ARM Ltd! How's StrongARM coming along?

> Am I to understand from this posting, that you are developing RO4
> independently from Acorn?

No, we don't wish to do this. Doing this would destroy the Acorn usership, with PD stuff being developed for Tornado and commercial stuff for RISC-OS.

No, what our problem with RISC-OS is that, since RISC-OS 2, there hasn't really been any /real/ improvement of the operating system. And with RO3, some of the supplied software (eg; Pinboard) is positively dire, and not only that the OS runs like a drain etc etc etc. RO3 was, in many peoples eyes, a disaster, mine included. I still use RO2 for example, never seeing why I should pay my hard-earned money for a hash of an operating system. We also believe Acorn are not putting enough resources into developing the correct areas of RISC-OS, and since they seem intent on developing a progressively worse and worse SharedCLibrary, and orientating all of the OS around it, we plan to do something about it. Even if this project never leaves the theoretical side, maybe it will make Acorn wake up and realise a lot of people are annoyed. Especially on fidonet, where the lack of development on Basic is really p***ing off a lot of people.

Look at it this way: the last rumours of dissent with Acorn's operating systems was in the days of Arthur. Now they are writing RISC-OS 4, and the same things are beginning to happen. Maybe it's about time they started writing an OS, starting again from RO2, which will really kick ass. Then again, maybe, and probably they won't. In which case, Tornado will be here.

Tornado is intended to function as an alternative development of the operating system from RO2 onwards, and I think you'll be impressed. It will be written mostly in C, bits of assembler and some of the demo stuff will be in Basic. It will remain compatible with current and future versions of RISC-OS, running alongside it rather than on top of it.

One of the handy things implemented by RO2 over Arthur was the very easy way it is to extend the existing OS. This will be used to its fullest extent.

I wonder what Acorn made of *that* reply? ;^)

The Finishing Touch

By James Larcombe (Wizard)
of DizzyWizard Software.
22-8-95

For those readers who didn't see the last (3rd.) issue of 'Frobnicate', this is the second in a series of articles by me which deal with what else to add to a Risc OS application when you think you've finished it.

Last issue I told you how to ensure that your users get the best set of icons possible, no matter what their monitor type. At the end, I promised that this issue would include a piece on iconising. However, I have subsequently changed my mind, as what that article would have dealt with is rather esoteric and of little practical use to many people. Instead I shall explain the purpose and value of international support, and the practicalities of using MessageTrans to achieve it.

What is international support, and why should I bother with it?

The phrase 'international support' means exactly what it says: making your application truly international. This has obvious advantages for both the programmer and the user. For the programmer, his application can gain a much wider user base outside of the United Kingdom. With the advent of the InterNet and other such means of electronic communication, geography is now practically irrelevant in the distribution of PD software. Commercial software vendors may also find international support boost sales considerably, and may find it worthwhile establishing distribution bases in a variety of countries.

Of course, the foreign user benefits too: he doesn't have to blunder along with the application, using whatever English he may know. After all, applications often have to talk in quite technical language, and even people knowing a fair smattering of English are still likely to have problems with it.

Acorn themselves have produced a German version of Risc OS, and have attempted to set up distribution networks for their computers abroad. Therefore the application writer must not get left behind. In this article I will attempt to explain what should be done to provide international support for your applications, and how it should be implemented.

How is international support implemented on the Arc?

Many of you will have heard of MessageTrans - it is a module which comes built in to Risc OS 3 (and can be loaded from disc for Risc OS 2). It's main purpose is to assist in international support. Indeed, the OS often uses it internally.

The basic principle of MessageTrans is tokenisation. This means that instead of 'hard-coding' a string into your code, it should be referred to by a token instead, and looked up from a separate file. This abstraction makes it far easier to change the language, since only one text file containing all the messages used by the application has to be updated, and the code doesn't have to be altered at all.

This file is usually called "Messages". I will make a few comments on a suitable location for this file later on, but needless to say it is located somewhere inside your <App$Dir>. The basic file format is as follows:

```
<token>:<string>
```

Example...

```
Err:An unexpected internal error has occurred.
```

You can also make more than one token match the same string, thus:

```
Err/Error:An unexpected internal error has occurred.
```

This will return the string if either “Err” or “Error” is looked up. The messages file can contain comments, and these should be prefixed with “#”. Tokens can contain wild cards which will match any character (”?”). I’ll detail a few more features of the file format later on, but it’s really quite simple.

Using MessageTrans

By now you are thinking “So... this messages file is very pretty, but how do I use it with my code?”. Well, that’s where the MessageTrans module comes in. First of all you need to open up your message file. I suggest you do this as part of your initialisation routine. You can either get MessageTrans to grab some RMA space for itself, or (more usually) you can get hold of some memory yourself and tell MessageTrans to use that. The BASIC below shows how to do the latter.

```
DEF PROCLoadMessages
  msgsname$="<App$Dir>.Messages"
  SYS "MessageTrans_FileInfo",,msgsname$ TO msgflags%,,msgsize%
  IF msgflags% AND 1 THEN msgsbuffer%=0 ELSE msgsbuffer%=FNalloc(msgsize%)
  msgsfiledesc%=FNalloc(17+LEN msgsname$)
  $(msgsfiledesc%+16)=msgsname$
  SYS "MessageTrans_OpenFile",msgsfiledesc%,msgsfiledesc%+16,msgsbuffer%
ENDPROC
```

In this example, FNalloc is assumed to be your memory-allocation routine. You could use a sliding heap manager for example, or a ‘clever’ RMA grabbing routine which avoids fragmentation. Anyway, you end up with a block of memory allocated for the file itself, and another block used for the descriptions, which is basically used as a file handle in all subsequent MessageTrans calls.

Two SWIs are used in the example, and they are described below. After looking at the registers returned from the SWIs it is pretty obvious what most of the code does. One thing to note is that if the messagefile happens to be already in memory (unlikely in most situations) then it is accessed directly from there.

```
                    MessageTrans_FileInfo  (&41500)

On entry:  R1 = filename

On exit :  R0 = flags (bit 0 set if held in memory already, ignore bits 1-31)
           R2 = size of buffer needed to hold file
```

```
                    MessageTrans_OpenFile  (&41501)

On entry:  R0 = 4-word data structure
           R1 = filename
           R2 = buffer to hold file data (0 to use RMA)

On exit :  ---
```

In your finalisation code, you should include something like this... so that all the memory is freed up:

```
   SYS "MessageTrans_CloseFile",msgsfiledesc%
   FNrelease(msgsbuffer%)
   FNrelease(msgsfiledesc%)
```

...where FNrelease is assumed to be a function that will release memory previously claimed. Below is the information for the new call...

Looking up a token

Now we've learnt how to open and close a messages file, we need to know how to access the file, and how to look up tokens. As I explained earlier, this is the basis of string substitution. So... here is a function which, when given a valid token, will refer to the messages file in memory and return the required string. Again, I'll go through the SWIs and how it works afterwards.

```
DEFFNmsgs_lookup(token$)
  LOCAL flags%,length%
  SYS "XMessageTrans_Lookup",msgsfiledesc%,token$,block%,256,"","","","" TO ,,,l
ength%;flags%
  IF flags% AND 1 THEN = token$
  block%?length%=13   : REM terminates it with [13]
=$block%
```

This assumes that you have defined a standard 256 byte block called 'block%', which you don't mind being overwritten by this routine. You can use the same block as you use for WIMP SWI calls, but it may be preferable to use a separate buffer. Also, you must have called the procedure I outlined earlier to open the messages file in the first place.

If the token is found in the messages file initialised earlier, then a string is returned. If the token could not be found, then the function returns the token that you gave it.

Here's the info for "MessageTrans_Lookup"...

```
                    MessageTrans_Lookup  (&41502)

On entry: R0 = 4-word data structure
          R1 = Token (terminated by any ctrl char, space, "," or ')' )
          R2 = buffer
          R3 = buffer size
          R4 = parameter 0
          R5 = parameter 1
          R6 = parameter 2
          R7 = parameter 3

On exit:  R1 = pointer to token-terminator
          R2 = result string
          R3 = size of result
```

Parameter substitution - a handy digression

In the SWI info above, you'll notice that the 4 registers detailed as being "parameter 0" etc. were set to "" in my above example. These registers are used for something called parameter substitution. This means that given up to 4 strings, we can substitute them for certain special parts of the message looked up. Here is, therefore, more file format info showing how to include parameter substitution.

```
ERR:An unexpected internal error has occurred (number %0)
POLL:Polling %0, telephone number %1
```

These examples are slightly contrived, but suffice to demonstrate the principle. Taking the first example, imagine in a hypothetical error-handler that we want to inform the user that something weird has happened, but at the same time we want to return a code useful for debugging. Supplying the error number in R4 on a call to MessageTrans_Lookup (with the token set to "ERR") would substitute this number for the '%0' part of the message itself. Therefore the error would read something like 'An unexpected internal error has occurred (number 158)'.

In the second example, we must imagine a fidonet mailer that wants to write a message to the log file detailing the address and telephone number of the place being polled. It would call MessageTrans_Lookup with the fido address in R4 and the phone number in R5, and these strings would get substituted for %0 and %1 respectively. Therefore an example may be: 'Polling 2:255/93.0, telephone number 01752261434'.

We now need a function that will lookup a message, and substitute the appropriate parameters. I saw recently a very clever way of doing this, and an example follows. [my thanks go to whoever did it originally... I think it was intended for use by anyone]

```
DEF FNmessage0(token$)           : = FNmessage4(token$, "", "", "", "")
DEF FNmessage1(token$,m1$)       : = FNmessage4(token$,m1$, "", "", "")
DEF FNmessage2(token$,m1$,m2$)   : = FNmessage4(token$,m1$,m2$, "", "")
DEF FNmessage3(token$,m1$,m2$,m3$) : = FNmessage4(token$,m1$,m2$,m3$, "")
DEF FNmessage4(token$,m1$,m2$,m3$,m4$)
 LOCAL flags%,length%
  SYS "XMessageTrans_Lookup",msgsfiledesc%,token$,block%,256,m1$,m2$,m3$,m4$ TO
,,,length%;flags%
  IF flags% AND 1 THEN = token$
  block%?length% = 13
=$block%
```

Do you see what is happening here? If we want to substitute no parameters, we call FNmessage0(token$). For 1 parameter, we call FNmessage1(token$,m1$). For 2, we call FNmessage2(token$,m1$,m2$) and so on, all the way up to 4. All of these 'interim' functions call FNmessage4, but will the unused parameters set to "".

This set of functions should suffice for any purpose. I'll explain some more important uses of parameter substitution later on, but needless to say it is a very powerful 'spin-off' of internation support. One thing to note is that obviously the strings you wish to substitute should NOT be words of a language, as this would defeat the object of MessageTrans and this article!

Anyway, getting back to our core topic of international support, we've now covered most of the mechanics of using MessageTrans. The module offers several other calls, which do come in handy sometimes, but these are the ones which are needed for international support. The others, whilst outside the scope of this article, are still worth investigating.

Uses of MessageTrans

So, we can now load up nationality-specific strings from a centralized resource. To give you a few ideas about what use this actually is, I'm going to give a few examples of the main uses of MessageTrans.

Firstly, I should say that MessageTrans should NOT be used to read in icon texts for templates. Templates are separated from the code already, so there isn't any need to fiddle around re-setting icon texts etc. from MessageTrans. What you should do is have a different Templates file for each different nationality supported by your application. More information on this later, in the section regarding structuring your internal application directory.

The main use of MessageTrans is for interactive help. Risc OS software SHOULD support interactive help, but it is surprising how many applications don't. However, yours should. If any one wants any help on how to give interactive help on windows and menus, then wait around: I might possibly consider this as a suitable subject for a future "Finishing Touch".

Anyway, when you support interactive help, rather than having the help text hard-coded into the program itself it is internationally better to load it up from the messages file. You may wish to number your tokens "H01","H02","H03" etc., where the number denotes the icon number that the token gives help on. That way you can rapidly and simply construct the correct token from the icon number, and lookup the correct string.

Parameter substitution can be of great help for complex, context-sensitive helptexts. As an example...

```
H01:This icon shows how much disc space is used for virtual memory. At present, %0 kilobytes are used. Click the right arrow to increase this figure, and the left arrow to decrease it.
```

A quick word about helptext - make it clear, but quite concise. Also, remember to include text for greyed out menus items, eg “This menu item is greyed out because there is no block selected” rather than “Move pointer right to save block” when you obviously can’t! This avoids confusing the novice user, and is nicer, anyway. Don’t overdo humour, either: if the user wanted a laugh he/she would be using a PC. *9-D

Other common uses of MessageTrans are error messages, log file text and prompts. There are many, many more: use your imagination! You can of course use all of this outside the desktop too, so the possibilities are endless.

Now I’m going to discuss some other issues of international support that are of considerable importance, especially now we can use MessageTrans. Firstly, there is the internal layout of your application.

Internal application directory structure

For an efficient international support system, it is important to structure your directories properly. This article isn’t really a suitable place to make comments about structuring general resources etc., but I will make a few notes about where to put nationality-specific things.

I would suggest that you have a directory inside your application called “Resources”. Inside this should go any general resources, but most importantly there should be further subdirectories, called for instance “UK”, or “Germany”. In each of these are the resources that need to be different for each country. I would suggest a minimum of the messages file and the templates, and possibly you might like to put other things in there too.

With a setup like this, your application could theoretically scan the “Resources” directory using OS_File, taking notice only of other directories, and thus build up a list of supported countries. This could then be used to provide a country-selection menu for the user, as in Arc-Binkley.

This is only a suggestion, and isn’t essential, but it seems to be to be a fairly efficient way of doing things.

Other international support issues

I’ve covered practically everything to do with messages files now, so it only remains to give a few words of advice about other international support matters. There is a useful (if short) section in the style guide relating to all this, and whilst I usually try to avoid duplicating that publication here, it is worth re-iterating some of the points raised.

First, avoid using icons that only really make sense if your first language in English. In other words, don’t use icons that rely on a certain pun, or play on words. Also, you should avoid icons which are only relevant in specific cultures. I can’t think of any examples of this, but make sure the metaphor works abroad as well as in England. Obviously, don’t use text in graphics if you can help it, since this will be trickier to replace.

Don’t make any assumptions regarding the character set or keyboard layout, and allow the use of ALT and top-bit set characters in your application. This means that people from other countries will be able to generate all their favourite accents etc., and will be able to use their national character set.

If possible, allow for a comma to be used in the place of the full stop when used in decimal point values. Make sure you use the system calls for dates etc., since that way the user will be able to configure his or her own preferred date style.

Finally, you might like to consider changing language-specific mnemonic key short-cuts. These are the ones where the mnemonic is the first letter of a word. These rarely work in other languages, unless you are lucky.

Conclusions

Hopefully you’ve read something in this article that has made you think about changing the way you do things. Indeed, I should expect so: I hardly ever see application handle international support properly. Not even the ‘big name’ apps all do, as some of you will have noticed.

One thing I’ve not touched upon is the translation itself : if you are not fluent in another language it is probably best to pay some sort of translation bureau to do the work for you. You should consider supporting most European countries, but obviously for commercial applications you will only need to support those countries in which you will be distributing your software.

Any queries with any of this, drop me a line and I’ll do my best. Thanks must at this stage go to Guttorm Vik for writing StrongHelp (where I learnt about the SWIs myself) and whoever wrote the clever bit of code I used in the parameter-substitution bit.

NEXT ISSUE: Any requests? Otherwise it could be interactive help, desktop boot files, DragASprite or indeed anything else that begins to interest me!

Written by James Larcombe.
Email: dizzywiz@digibank.demon.co.uk
Fidonet: 2:255/93.4

The big ©

The question of software copyright

The laws regarding copyright affect everybody: whether you write a really simple program, a complex program or if you just use other people's software.

Have you, though, ever wondered what happens when you slap a © on your work – or indeed if you are entitled to? What about your rights regarding other people's work? Can you 'hack' a program?

Well, have a read of this. It should pretty much clarify the situation in the EC.

[only in law can you get away with stating a million paragraphs with the word "Whereas"!]

No L 122/42
Official Journal of the European Communities 17.5.91

II

(Act whose publication is not obligatory)

COUNCIL DIRECTIVE
of 14 May 1991
on the legal protection of computer programs

(91/250/EEC)

THE COUNCIL OF THE EUROPEAN COMMUNITIES,

Having regard to the Treaty establishing the European Economic Community and in particular Article 100a thereof,

Having regard to the proposal from the Commission (1),

In cooperation with the European Parliament (2),

Having regard to the opinion of the Economic and Social Committee (3),

Whereas computer programs are at present not clearly protected in all Member States by existing legislation and such protection, where it exists, has different attributes;

Whereas the development of computer programs requires the investment of considerable human, technical and financial resources while computer programs can be copied at a fraction of the cost needed to develop them independently;

Whereas computer programs are playing an increasingly important role in a broad range of industries and computer program technology can accordingly be considered as being of fundamental importance for the Community's industrial development;

Whereas certain differences in the legal protection of computer programs offered by the laws of the Member States have direct and negative effects on the functioning of the common market as regards computer programs and such differences could well become greater as Member States introduce new legislation on this subject;

Whereas existing differences having such effects need to be removed and new ones prevented from arising, while differences not adversely affecting the functioning of the common market to a substantial degree need not be removed or prevented from arising;

Whereas the Community's legal framework on the protection of computer programs can accordingly in the first instance be limited to establishing that Member States should accord protection to computer programs under copyright law as literary works and, further, to establishing who and what should be protected, the exclusive rights on which protected persons should be able to rely in order to authorize or prohibit certain acts and for how long the protection should apply;

Whereas, for the purpose of this Directive, the term 'computer program' shall include programs in any form, including those which are incorporated into hardware; whereas this term also includes preparatory design work leading to the development of a computer program provided that the nature of the preparatory work is such that a computer program can result from it at a later stage;

Whereas, in respect of the criteria to be applied in determining whether or not a computer program is an original work, no tests as to the qualitative or aesthetic merits of the program should be applied;

Whereas the Community is fully committed to the promotion of international standardization;

Whereas the function of a computer program is to communicate and work together with other components of a computer system and with users and, for this purpose, a logical and, where appropriate, physical interconnection and interaction is required to permit all elements of software and hardware to work with other software and hardware and with users in all the ways in which they are intended to function;

Whereas the parts of the program which provide for such interconnection and interaction between elements of software and hardware are generally known as ‘interfaces’;

Whereas this functional interconnection and interaction is generally known as ‘interoperability’; whereas such interoperability can be defined as the ability to exchange information and mutually to use the information which has been exchanged;

Whereas, for the avoidance of doubt, it has to be made clear that only the expression of a computer program is protected and that ideas and principles which underlie any element of a program, including those which underlie its interfaces, are not protected by copyright under this Directive;

Whereas, in accordance with this principle of copyright, to the extent that logic, algorithms and programming languages comprise ideas and principles, those ideas and principles are not protected under this Directive;

Whereas, in accordance with the legislation and jurisprudence of the Member States and the international copyright conventions, the expression of those ideas and principles is to be protected by copyright;

Whereas, for the purposes of this Directive, the term ‘rental’ means the making available for use, for a limited period of time and for profit-making purposes, of a computer program or a copy thereof; whereas this term does not include public lending, which, accordingly, remains outside the scope of this Directive;

Whereas the exclusive rights of the author to prevent the unauthorized reproduction of his work have to be subject to a limited exception in the case of a computer program to allow the reproduction technically necessary for the use of that program by the lawful acquirer;

Whereas this means that the acts of loading and running necessary for the use of a copy of a program which has been lawfully acquired, and the act of correction of its errors, may not be prohibited by contract; whereas, in the absence of specific contractual provisions, including when a copy of the program has been sold, any other act necessary for the use of the copy of a program may be performed in accordance with its intended purpose by a lawful acquirer of that copy;

Whereas a person having a right to use a computer program should not be prevented from performing acts necessary to observe, study or test the functioning of the program, provided that these acts do not infringe the copyright in the program;

Whereas the unauthorized reproduction, translation, adaptation or transformation of the form of the code in which a copy of a computer program has been made available constitutes an infringement of the exclusive rights of the author;

Whereas, nevertheless, circumstances may exist when such a reproduction of the code and translation of its form within the meaning of Article 4 (a) and (b) are indispensable to obtain the necessary information to achieve the interoperability of an independently created program with other programs;

Whereas it has therefore to be considered that in these limited circumstances only, performance of the acts of reproduction and translation by or on behalf of a person having a right to use a copy of the program is legitimate and compatible with fair practice and must therefore be deemed not to require the authorization of the rightholder;

Whereas an objective of this exception is to make it possible to connect all components of a computer system, including those of different manufacturers, so that they can work together;

Whereas such an exception to the author’s exclusive rights may not be used in a way which prejudices the legitimate interests of the rightholder or which conflicts with a normal exploitation of the program;

Whereas, in order to remain in accordance with the provisions of the Berne Convention for the Protection of Literary and Artistic Works, the term of protection should be the life of the author and fifty years from the first of January of the year following the year of his death or, in the case of an anonymous or pseudonymous work, 50 years from the first of January of the year following the year in which the work is first published;

Whereas protection of computer programs under copyright laws should be without prejudice to the application, in appropriate cases, of other forms of protection; whereas, however, any contractual provisions contrary to Article 6 or to the exceptions provided for in Article 5 (2) and (3) should be null and void;

Whereas the provisions of this Directive are without prejudice to the application of the competition rules under Articles 85 and 86 of the Treaty if a dominant supplier refuses to make information available which is necessary for interoperability as defined in this Directive;

Whereas the provisions of this Directive should be without prejudice to specific requirements of Community law already enacted in respect of the publication of interfaces in the telecommunications sector or Council Decisions relating to standardization in the field of information technology and telecommunication;

Whereas this Directive does not affect derogations provided for under national legislation in accordance with the Berne Convention on points not covered by this Directive, HAS ADOPTED THIS DIRECTIVE:

Article I

Object of protection

1. In accordance with the provisions of this Directive, Member States shall protect computer programs, by copyright, as literary works within the meaning of the Berne Convention for the Protection of Literary and Artistic Works. For the purposes of this Directive, the term ‘computer programs’ shall include their preparatory design material.

2. Protection in accordance with this Directive shall apply to the expression in any form of a computer program. Ideas and principles which underlie any element of a computer program, including those which underlie its interfaces, are not protected by copyright under this Directive.

3. A computer program shall be protected if it is original in the sense that it is the author’s own intellectual creation. No other criteria shall be applied to determine its eligibility for protection.

Article 2

Authorship of computer programs

1. The author of a computer program shall be the natural person or group of natural persons who has created the program or, where the legislation of the Member State permits, the legal person designated as the rightholder by that legislation. Where collective works are recognized by the legislation of a Member State, the person considered by the legislation of the Member State to have created the work shall be deemed to be its author.

2. In respect of a computer program created by a group of natural persons jointly, the exclusive rights shall be owned jointly.

3. Where a computer program is created by an employee in the execution of his duties or following the instructions given by his employer, the employer exclusively shall be entitled to exercise all economic rights in the program so created, unless otherwise provided by contract.

Article 3

Beneficiaries of protection

Protection shall be granted to all natural or legal persons eligible under national copyright legislation as applied to literary works.

Article 4

Restricted Acts

Subject to the provisions of Articles 5 and 6, the exclusive rights of the rightholder within the meaning of Article 2, shall include the right to do or to authorize:

(a) the permanent or temporary reproduction of a computer program by any means and in any form, in part or in whole. Insofar as loading, displaying, running, transmission or storage of the computer program necessitate such reproduction, such acts shall be subject to authorization by the rightholder;

(b) the translation, adaptation, arrangement and any other alteration of a computer program and the reproduction of the results thereof, without prejudice to the rights of the person who alters the program;

(c) any form of distribution to the public, including the rental, of the original computer program or of copies thereof. The first sale in the Community of a copy of a program by the rightholder or with his consent shall exhaust the distribution right within the Community of that copy, with the exception of the right to control further rental of the program or a copy thereof.

Article 5

Exceptions to the restricted acts

1. In the absence of specific contractual provisions, the acts referred to in Article 4 (a) and (b) shall not require authorization by the rightholder where they are necessary for the use of the computer program by the lawful acquirer in accordance with its intended purpose, including for error correction.

2. The making of a back-up copy by a person having a right to use the computer program may not be prevented by contract insofar as it is necessary for that use.

3. The person having a right to use a copy of a computer program shall be entitled, without the authorization of the rightholder, to observe, study or test the functioning of the program in order to determine the ideas and principles which underlie any element of the program if he does so while performing any of the acts of loading, displaying, running, transmitting or storing the program which he is entitled to do.

Article 6

Decompilation

1. The authorization of the rightholder shall not be required where reproduction of the code and translation of its form within the meaning of Article 4 (a) and (b) are indispensable to obtain the information necessary to achieve the interoperability of an independently created computer program with other programs, provided that the following conditions are met:

(a) these acts are performed by the licensee or by another person having a right to use a copy of a program, or on their behalf by a person authorized to to so;

(b) the information necessary to achieve interoperability has not previously been readily available to the persons referred to in subparagraph (a); and

(c) these acts are confined to the parts of the original program vhich are necessary to achieve interoperability.

2. The provisions of paragraph 1 shall not permit the information obtained through its application:

(a) to be used for goals other than to achieve the interoperability of the independently created computer program;

(b) to be given to others, except when necessary for the interoperability of the independently created computer program; or

(c) to be used for the development, production or marketing of a computer program substantially similar in its expression, or for any other act which infringes copyright.

3. In accordance with the provisions of the Berne Convention for the protection of Literary and Artistic Works, the provisions of this Article may not be interpreted in such a way as to allow its application to be used in a manner which unreasonably prejudices the right holder’s legitimate interests or conflicts with a normal exploitation of the computer program.

Article 7

Special measures of protection

1. Without prejudice to the provisions of Articles 4, 5 and 6, Member States shall provide, in accordance with their national legislation, appropriate remedies against a person committing any of the acts listed in subparagraphs (a), (b) and (c) below:

(a) any act of putting into circulation a copy of a computer program knowing, or having reason to believe, that it is an infringing copy;

(b) the possession, for commercial purposes, of a copy of a computer program knowing, or having reason to believe, that it is an infringing copy;

(c) any act of putting into circulation, or the possession for commercial purposes of, any means the sole intended purpose of which is to facilitate the unauthorized removal or circumvention of any technical device which may have been applied to protect a computer program.

2. Any infringing copy of a computer program shall be liable to seizure in accordance with the legislation of the Member State concerned.

3. Member States may provide for the seizure of any means referred to in paragraph 1 (c).

Article 8

Term of protection

1. Protection shall be granted for the life of the author and for fifty years after his death or after the death of the last surviving author; where the computer program is an anonymous or pseudonymous work, or where a legal person is designated as the author by national legislation in accordance with Article 2 (1), the term of protection shall be fifty years from the time that the computer program is first lawfully made available to the public. The term of protection shall be deemed to begin on the first of January of the year following the abovementioned events.

2. Member States which already have a term of protection longer than that provided for in paragraph I are allowed to maintain their present term until such time as the term of protection for copyright works is harmonized by Community law in a more general way.

Article 9

Continued application of other legal provisions

1. The provisions of this Directive shall be without prejudice to any other legal provisions such as those concerning patent rights, trade-marks, unfair competition, trade secrets, protection of semi-conductor products or the law of contract. Any contractual provisions contrary to Article 6 or to the exceptions provided for in Article 5 (2) and (3) shall be null and void.

2. The provisions of this Directive shall apply also to programs created before 1 January 1993 without prejudice to any acts concluded and rights acquired before that date.

Article 10

Final provisions

1. Member States shall bring into force the laws, regulations and administrative provisions necessary to comply with this Directive before 1 January 1993.

When Member States adopt these measures, the latter shall contain a reference to this Directive or shall be accompanied by such reference on the occasion of their official publication. The methods of making such a reference shall be laid down by the Member States.

2. Member States shall communicate to the Commission the provisions of national law which they adopt in the field governed by this Directive.

Article 11

This Directive is addressed to the Member States.

Done at Brussels, 14 May 1991.

For the Council

The President

J. F. POOS

References:

(1) OJ No C 91, 12. 4. 1989, p. 4; and OJ No C 320, 20. 12. 1990, p. 22.
(2) OJ No C 231, 17. 9. 1990, p. 78; and Decision of 17 April 1991, yet published in the Official Journal).
(3) OJ No C 329, 30. 12. 1989, p. 4.

Your “original works” are automatically covered by copyright as you have created it. However it is best to make totally sure by expressing the copyright in your programs. The typical convention is:
<copyright symbol> Copyright <your name> <date>

Eg:
© Copyright Richard Murray 1994.

You *need* the copyright symbol as certain countries may not understand the meaning of “copyright” (in English). Technically, also the convention of “(C)” is not strictly correct. But on the Acorn we need not worry. Our character set has a copyright character (ASCII 169) which can be keyed with Shift+Alt+C.

You can make small variations on the theme if you are releasing various different versions in a year. For example in my programs I would use:
© Richard Murray 2nd October 1995
which helps me also to know which version is which.

A Light In The Dark (found on TWoC)

For years it has been believed that electric bulbs emit light. However, recent information from Bell Labs has proven otherwise. Electric bulbs do not emit light, they suck dark. Thus they are now called dark suckers. The Dark Sucker Theory, according to a Bell Labs spokesperson, proves the existence of dark, that dark has a mass heavier than that of light, and that dark travels faster than light.

The basis of the Dark Sucker Theory is that electric bulbs suck dark. Take for example the dark suckers in the room where you are. There is less dark in the immediate area of the dark suckers than there is elsewhere in the room. The larger the dark sucker, the greater its capacity to suck dark. Dark suckers in a parking lot have a much greater capacity than the ones in this room. As with all things, dark suckers don’t last forever. Once they are full of dark they can no longer suck. This is proven by the black spot on a full dark sucker. A candle is a primitive dark sucker. A new candle has a white wick. You will notice that, after the first use, the wick turns black -- representing all the dark which has been sucked into it. If you hold a pencil next to the wick of an operating candle, the tip will turn black because it got in the way of the dark flowing into the candle.

Unfortunately, these primitive dark suckers have a very limited range. There are, fortunately, portable dark suckers. The bulbs in these cannot handle all of the dark by themselves, and require the use of additional dark storage units. When the dark storage unit, referred to by some as a battery, is full it must either be emptied or replaced before the portable dark sucker can operate again.

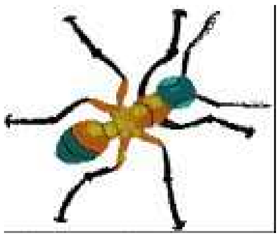
Dark has mass. When dark goes into a dark sucker, friction from this mass generates heat. Thus it is not wise to touch an operating dark sucker. Candles present a special hazard because the dark must travel in the solid wick instead of through glass. This generates a large quantity of heat, which makes it inadvisable to touch an operating candle.

Dark is also heavier than light. If you swim deeper and deeper you notice that it slowly gets darker and darker. When you reach a depth of approximately 80 meters, you are in total darkness. This is because the heavier dark sinks to the bottom of the water and the lighter light floats to the top. The immense power of dark can be utilized to humankind’s advantage. Dark which has settled to the bottoms of lakes can be pushed through turbines to generate electricity. In this way dark can be forced into the oceans where it can be safely stored.

Prior to the invention of the turbine it was much more difficult to get dark from rivers and lakes to the oceans. The Indians recognized this problem and tried to solve it. When on a river in a canoe travelling in the same direction as the flow of dark, Indians paddled slowly, so as not to stop the flow of dark. When they travelled against the flow of dark they paddled quickly to help push the dark along its way.

Finally, it becomes clear that dark is faster than light. If you stand in an illuminated room in front of a closed, dark closet you notice that, as you slowly open the closet door, light slowly enters the closet. However the dark moves so quickly that you are not able to see the dark leave the closet.

In conclusion, scientists from the Bell Labs have noted that dark suckers make our lives easier and more enjoyable. So the next time you look at an electric bulb remember that its function is actually that of a dark sucker.



ANTS ! ! !

Ants

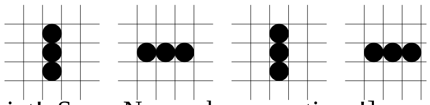
by Nava Whiteford

No, I’ve not gone completely mad!!! (That happened long ago).

I’m not talking about the kind of Ant that gets into your picnic basket and messes up you food. I’m talking about a different type of Ant – a computerised Ant.

About 20 years ago a man call John Conway devised a set of rules call Life. Since then, Life has been coded and re-coded for just about every computer. So most users have heard of it. This type of “game” or set of rules is called cellular automata. Now basically, this means that you have got lots of cells (in life each animal is a cell) which move about according to a set of rules.
The odd thing about cellular automata is that with very relatively few simple rules you get a very complex/chaotic result (e.g In Life when you put down a couple of cells and get a big blob after a few generations). Another odd thing is that the only way to find out what you will get after a certain number of generations is to go through the generations. There is no formula that will give you the answer. This makes cellular automata interesting (and annoying) for physicists who are trying to find a TOE (or Theory of Everything) in fact it messes them up quite a bit I mean if they can’t figure out this computer simulation out how have they got any hope of doing it with something like the universe?

[editors note: In fact, certain automa patterns have a predictable outcome, for example, the diagram shows a “blinker”. But this is very beside-the-point! Sorry Nava, please continue!]



Anyway, people tend to think that the problem with Life is that the rules are too complex and the big mess you get can be justified by these rules. So some bloke came up with Ants. And this is what the remainder of this article is about.

The ant(s) live on a grid. Each cell of the grid can be black or white. To start with our cell are going to be black but if you like you can have yours white, it doesn’t really matter. The rules for Ants are much simpler than for Life.

Here they are:

1. When an ant lands on a cell the colour of that cell is flipped B to W, W to B
2. The ant moves one cell per generation it cannot move diagonally.
3. If the ant lands on a white cell it turns left. If it lands on a black cell it turns right.

Nice and simple. And the code to do that is only a few lines long:

```
1  for(;;)
2  {
3    colour=os_point(antx,anty);
4    if(colour==0) {os_gcol(0 ,1); heading++; if(heading== 4) heading=0; }
5    if(colour==1) {os_gcol(0 ,0); heading--; if(heading==-1) heading=3; }
6    os_rectanglefill(antx, anty, 1, 1);
7
8    if(heading==0) anty++;
9    if(heading==1) antx++;
10   if(heading==2) anty--;
11   if(heading==3) anty++;
12 }
```

Not too complex is it. (A BASIC version is also supplied)

Okay, let’s have a look at this then...

Right, lines 1-3 are quite simple. Line 3 just tells us the colour of the cell where that ant is.
Line 4 changes the colour to be put on to that cell flipping it from black to white. Line 5 does this the other way round.
Line 4 then says add 1 to the heading (heading++;) Then says if heading is 4 then heading = 0 this loops it back round again.

The headings work like so:

```
 N
 .
/ \
 |  Up the screen is North.
 |
```

When going North heading = 0
When going East heading = 1
When going South heading = 2
When going West heading = 3

You can see that if you add one to the heading you will then turn right and if you take one from the heading you will then turn left. Then if heading is 0 and if heading is 3 are just to make it loop round to the beginning again.

The next bit Lines 8-11 just make the ant go in the direction of the heading.

So we have got a simple one–ant program.

From that we can make a simple multi–ant program and mess about with it in other ways (See “colourful” for some fun :-)).

Things can get very odd with ants (Take a look at multi–ant with the initial settings). There are lots of thing I haven’t covered (highways, diamonds etc.) and maybe will do in follow-up article (Richard?) [Editors note: Got for it!]. Maybe once you’ve been given a chance to mess about with the programs and make up your own patterns. Then we could shove some of your stuff in the article too. Send your stuff to me on the DigiBank.

A MultiAnt program, a Single ant program„ and a colourful version are all in the Ant.Progs dir. There is also a BASIC program in the Prog dir. All the c source is in the Ants dir. Some Info about the programs are in the ReadMe and Progs.ProgInfo files.

Oh, and try these settings on MultiAnt

2 ants

ant 0 x = 70
ant 0 y = 70
ant 0 heading = 3

ant 1 x = 73
ant 1 y = 70
ant 1 heading = 3

Good, err?

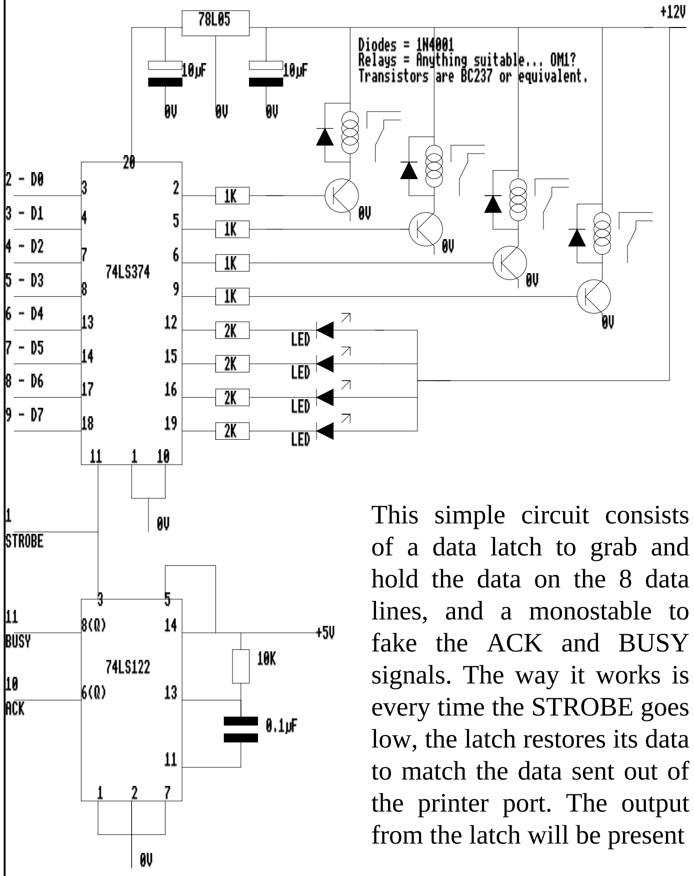
Nava Whiteford (edited by Richard Murray)

In a first for Frobnicate, you’ll find this stuff in the “Ants” subdirectory of the Frobnicate archive.

Many thanks to Nava for this. These programs are those things you see on Acorn User and think “Oh god, not another silly graphic thing”. Amazing how I managed to spend two hours staring at my screen watching these blocks move. A hint for those of you with the BASIC version, shove an “A=GET” in the loop so you can see each move at a time.

[wow: Frobnicate publishes something intellectual. What’s going on!? Hey, yey, yey, yeh-eh – hey, yey, yey. I said “Hey, what’s going on?”. :-)]

Centronics Control Interface



This simple circuit consists of a data latch to grab and hold the data on the 8 data lines, and a monostable to fake the ACK and BUSY signals. The way it works is every time the STROBE goes low, the latch restores its data to match the data sent out of the printer port. The output from the latch will be present even when the data on the printer port disappears. The rising edge of the STROBE triggers the monostable which sends back the BUSY and ACK signals (so the computer knows that that data was received okay). The 78L05 is a 5V regulator to power the ICs. There are two types of output, four status LEDs and four relays. You could pick’n’mix. It’s a simple circuit. The transistors buffer the relays to the latch, the diodes stop EMF spikes from melting the transistors.
The relays are turned on by sending a high bit to the latch. The LEDs, in reverse, are turned off by sending a high bit. So to turn on LED 1, 2 and 4 and active relays 2 and 3, you would send “00100110” which is ASCII character 37 or “%”. Play around with it. You’ll soon see.
Note: Don’t forget to suppress the linefeeds when you send the data. In BASIC, place a semicolon after a print statement, or send directly to the printer.

Here, the system is arranged thus:

Relay 1	= Controls cine camera frame release (animation mode)
Relay 2	= Focus stepper motor pulse
Relay 3	= Focus stepper motor direction
Relay 4	= Light switch
LED 1	= Frame will be taken
LED 2	= Focus telephoto
LED 3	= Focus wide
LED 4	= Lights on

Enjoy!!!

NOT SO Easy PeeC

The general consensus regarding Easy PeeC was "take a hike!!!". So I figured I might try something a little more 'techie'.

Firstly, we'll have a little bit of fun. The on-line pages of *tunefs*, like all Berkeley commands, ends with a 'Bugs' section. In this case it read:

```
Bugs:
This program should work on mounted
and active file systems, but it
doesn't. Because the superblock is
not kept on the buffer cache, the
program will only take effect if it
is run on dismounted  file systems;
if run on the root file system, the
system must be rebooted.
You can tune a file system, but you
can't tune a fish.
```

Honest! But you won't find this anymore. When people moves to SVr4 UNIX, the "Bugs" section was renamed "Notes" (as if that fools anybody) and all the sense of humour was removed – that's a great shame as C and UNIX require a huge sense of humour in order to be used effectively.

One thing I'm sure you didn't know was that an amount of C was actually written for the benefit of the compiler writer, not the programmer. Examples:

• Arrays start at 0 instead of 1, so in an array dimensioned to [100], writing data to [100] could case a crash as the array actually goes [0]...[99].
• The main C types map onto underlying hardware. C didn't support FP ops until the computer itself did.
• The `auto` keyword is useless. You get it by default so why.......
• Array names "decay" into pointers. It makes life easier to treat arrays as pointers instead of writing complex routines to handle arrays. But arrays and pointers are *not* the same...
• No nested functions. This makes compilation easier.
• The `register` keyword. This gave the compiler writer an idea about which variables the programmer expected to be most frequently referenced, and hence could be kept in registers. This is a waste of time and you get better code if the compiler allocates registers for individual uses of a variable rather than reserving them. The `register` keyword simplifies a compiler by giving this headache to the programmer.

Since this column is short this issue, I'll leave you with this:

The suggestion of undefined instructions causing your computer to explode isn't as far fetched as it might seem. The original IBM PC monitor operated at a horizontal scan rate supplied by the video controller. The flyback transformer relied on this being a reasonable frequency. However, it was possible for the software to set the video scan rate to zero, thus feeding a constant voltage into the primary winding of the flyback transformer. This then acted like a resistor and dissipated its power as heat instead of sending its power to the screen. This burned out the monitor in seconds.

There are also rumours that it is possible to burn out a BBC micro with a simple looping routine to toggle all the address and data lines on and off as quickly as possible.

Well, at least this issue's *Not so* Easy PeeC was at least readable by regular human beings. :-)

Analysis of Frobnicate

The results for this article were compiled on 28th August 1995. It is meant as an indication.

RESULTS FOR FROBNICATE 1, 2 & 3:

Arcade BBS:	
Ovation	7 / 16 / 10
Text & graphics	43 / 35 / 27
Impression	X / X / 12
ArcTic BBS:	
Ovation	X / 3 / X
Text & graphics	X / 4 / 3
Impression	X / X / 0
Digital Databank BBS:	
Ovation	2 / 10 / 0
Text & graphics	14 / 6 / 2
Impression	X / X / 4
Plasma Sphere & Northern ARM BBS's:	
Ovation	7 / 1 / X
Text & graphics	X / 0 / X
Impression	X / X / X
TOTALS:	
Ovation	16 / 30 / 10
Text & graphics	57 / 45 / 32
Impression	X / X / 16
Total inclusive	**73 / 75 / 58**

General Synopsis for 3rd October 1995:

I feel Frobnicate is doing fairly well. I started this project in order to see how viable it is to create a magazine and pass it around. It appears Frobnicate has between 60 and 80 readers. As Frobnicate is not printed and sold, it'll never match AU or Archive or RiscUser – but I'm happy with my current readership. Some more feedback would be nice though. :-)

Many many many many thanks go to Niall Douglas, James Larcombe and Nava Whiteford. Without your input Frobnicate would surely have died in the middle of issue 3.

For getting me this far, I'd like to also thank: Hugo Fiennes, John Stonier, DaviD Dade, David Coleman, Steve Pursey, Graham Cant, Thomas Olsson, Hans Ringdahl, Ricky Sarge, Helen Rayner, Keith Hall, Robin Abecasis, Graeme Read, Andrew Lobel (yup, I'm mad!), Daniel Aston, Rick Crook, Dane Koekoek, Robbie Record, Nick Hutton, Keith Marlow, Jp, Chris Jackson, Roy Moore, everybody at Acorn User, Charlie Bayliss, Acorn Computers, Bridget Fonda, Tetley, Dire Straits, Stephen King, Azathoth, Shumart! and Sambo. Anybody I forgot?

Finally, a touchy point.
Thank you, Loretta, for interesting chats...

I am disappointed by the return of the Reader Survey. I received only 6, with is 10%. All the rest of you – if you want this magazine to try to please you more – please send back your user survey. I *can* take criticism you know! ;^)

Here are some ideas in the thinking pot:

- Simple DIY projects.
- Info on 'interesting' Internet utilities.
- Interesting WWW/ftp sites.
- Info on the PD/ShareWare/Comms scene.

If you'd some software reviewed, please contact me. I'll be happy to kill^H^H^H^Hreview anything that will fit in under 1Mb of disc space [I've only 5Mb freespace. :-((( ].

Reader's Letters

Send your letters to:
"Richard Murray" at 2:254/86.1 (FidoNet) or "rmurray@digibank.demon.co.uk".

`When you you plan on releasing an Impression version of Frobnicate?`

Around about now I think. :-)

`Do you plan to release Frobnicate in a PC compatible format, for example GIFs and linefeed-terminated text in a PKZip archive?`

Tell me... Why would a person writing this magazine to target “techie” ***Acorn*** users wish to support PCs? It’s about time you PC people could read Acorn sprites via a converter. :-) After all, as I am trying to use more DrawFiles - how do you anticipate I convert them to your equivalent MetaFile?

`I am an Amiga user and...`

...sorry dude, wrong mag. Byeee!!!

FONTS

This short message will detail the fonts used in Frobnicate, and where you can obtain them from...

ANSI
This font is from Gareth Boden’s excellent TrueANSI program. You can use the PD font “MDA1” as a replacement.

CORPUS
This font is built into RiscOS 3.

HOMERTON
This font is built into RiscOS 3.

KEYS1000
This is rarely used in Frobnicate now, it’s available from good BBSs.

SELWYN
This font is used for the ’pointers’ to the topics on the front cover. It can be ignored if you don’t have it. Selwyn is supplied with the Acorn DTP package.

<system font>
This font is sometimes used in DrawFiles. Your draw module will render this correctly for you.

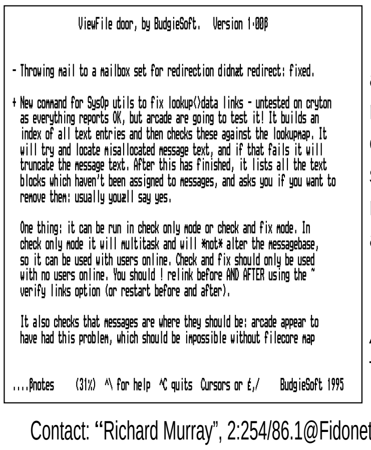
SYSTEM.FIXED
This font is available from good BBSs.

TRINITY
This font is built into RiscOS 3.

I hope that makes your font hunting easier...

That’s about it for the mailbag. <sigh>...

SysOps...



```
              ViewFile door, by BudgieSoft.   Version 1·00ß


- Throwing mail to a mailbox set for redirection didnæt redirect: fixed.

+ New command for SysOp utils to fix lookup()data links - untested on cryton
  as everything reports OK, but arcade are going to test it! It builds an
  index of all text entries and then checks these against the lookupmap. It
  will try and locate misallocated message text, and if that fails it will
  truncate the message text. After this has finished, it lists all the text
  blocks which haven’t been assigned to messages, and asks you if you want to
  remove them: usually youæll say yes.

  One thing: it can be run in check only mode or check and fix mode. In
  check only mode it will multitask and will *not* alter the messagebase,
  so it can be used with users online. Check and fix should only be used
  with no users online. You should ! relink before AND AFTER using the ~
  verify links option (or restart before and after).

  It also checks that messages are where they should be: arcade appear to
  have had this problem, which should be impossible without filecore map


....ßnotes    (31%)  ^\ for help  ^C quits  Cursors or £,/    BudgieSoft 1995
```

Have you ever wished for a utility to allow you to read the file(s) of your choice remotely? If so – take a look at the new demo, ViewFile, available from good BBSs.

Another ***free*** program from BudgieSoft.

Contact: “Richard Murray”, 2:254/86.1@Fidonet or “rmurray@digibank.demon.co.uk”.....

How come *they* have it – and *we* don’t???

Just a little box of cereal I found in France – and liked.

KELLOGGS???

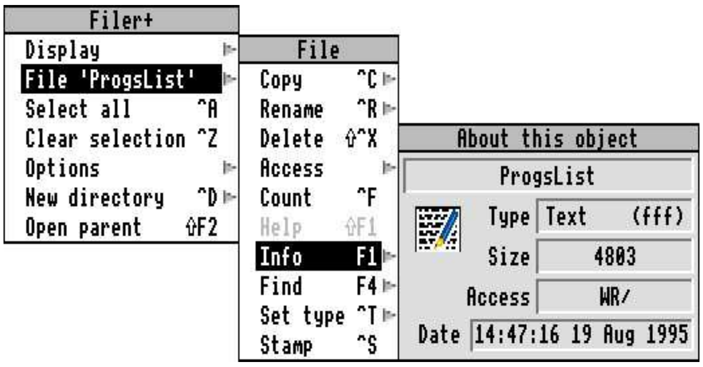
Feetch, feetch!

One of the sheer delights of the Acorn system is the filer. It can do what many other GUIs hiccup and wheeze over; stuff like copying two files from archive to RAMdisc, datestamping two directories full of files and formatting a floppydisc – *at the same time*! Apart from odd quirks, like Dismount and Format being next to each other – the filer is very good. However, it can be improved yet further...

Filer+ (by Jens H. Ovesen) adds these features:

- Hidden objects - just like DOS. Also the ability to super-hide a file so it stays really hidden.
- Local directory options (so your settings of Small Icons/Show All could be 'local' to that directory and not affect others.
- A proper SetType menu that lists all configured filetypes – and can bung in a sprite for good measure.
- Keyboard short-cuts for filer operations.
- Ability to define icons to be shown instead of the normal "directory folder" icon. Therefore you can have "Comms" show a modem, "DTP" show a scroll and "Games" show something like a Lemming. All other (undefined) directories will have the normal icon.
- Double-click-hold to imitate a Shift+double-click (load file into !Edit).

The patch is for RiscOS3.10 and Filer 1.64. It is easy to install and set up.



ProgsList
Type Text (fff)
Size 4803
Access WR/
Date 14:47:16 19 Aug 1995

This utility is highly recommended, and can be found on virtually any BBS.

Filer+ - 90% (!)

Rob the Slob, aka Robin Abecasis, has released a new version of his popular ArcQuoter program. This version is ArcBBS-friendly, so now you can ALL use Robin's door.

The purpose of ArcQuoter is threefold:

- Inserts a random quote into a text input window (such as fidomail). This is the main purpose of the program, so facilities for this are extensive.
- Inserts your "signature" into a text input window, useful for e–mail.
- Will spool files dragged to it to a text input window. Useful for shoving stuff across to somebody via modem (eg: in a chat). A nice touch is the hourglass shows the %age of the file sent.

The main alternative to this program is "YouWhat2" by Daniel Aston. However it appears that Daniel has left the comms scene, and may not be updating his version. This – on the other hand – offers many good features and is fairly easy to set up. The only gripe I have is the need to bother with this type of program at all... but that's hardly Robin's fault. :-)

ArcQuoter - **80%**

PS: Robin... Where's that AI quoter you promised?
[this is a private joke... ;^)]

Reviewed unbiasedly (hmmm!?)
by Richard Murray on a patched A5000
4Mb RAM, 40Mb IDE + 40Mb IDE + 1Gb SCSI,
SP_Dual issue 2, Vision, RiscOS 3.10 + patches,
Ground Control Teletext adaptor, AKF12 monitor,
150Mb tape streamer, USR Courier 14400, Pace
MicroLin 14400.

I hope soon to bring in-depth reviews of software
such as TrueANSI2, Commotion and ArmBBS.
Hey guys...?

Qu'est-ce que c'est, ça?

Never one to worry about being controversial (who? meee?), I'm going to prophesise about how wonderful and innovative Acorn technology is – and how come this technology is not appearing everywhere like PCs. In fact, Acorn is relatively unknown. Ask Joe-the-PC-dealer and he'll usually either look at you blankly or tell you that you should invest in an "industry standard" computer. The few that do know are just that.... a few.

But why? Do Acorn have a naff and overprices operating system? NO. Do Acorn make a big deal out of the latest 16/32bit CISC processor (that is code-compatible with an old 8-bit processor that was around in the late '70s – nearly 20 years ago!)? NO. People moan about Acorn having "a built-in legacy". Okay. So the operating system until recently had MODEs, and still has FXs. It is nothing like a certain type of computer that features:

- 8-bit processor compatible
- Absurd memory limits without bodges to get around them
- A complicated command-line based operating system.....
- .....to which a slightly dismal GUI system can be added.

What's so bad about legacies anyhow? The latest thing on the programming front is C++. Let me tell you, C++ developed from three angles:

- From C (1983-1989); derived from B (1970) and BCPL (1965)
- From Simula67 (1967)
- From ADA

So you can see that while C++ is fairly new, it's inspiration dates back into the dawn of (computer) time.

Back to the original question. Acorn have managed to successfully interface an Intel^{TM} CISC processor into their own RISC system. Sure, you can't run PageMaker^{TM} pseudo-native in the DeskTop...yet...but it's one heck of an achievement. Acorn were once *up-there* with the freaky innovation (who has never seen the underside of a BBC micro?) but now Acorn are no longer *there*. If we were, we'd have a computer system offering:

- Nice small RISC processor running at a comfortable 60MHz and outrunning the fastest PC
- 16Mb RAM fast enough to be directly accessed by the processor
- Screen modes up to 4000x4000 at 24bit, flicker free.

Sadly, not a lot has happened. Acorn seem to be concentrating elsewhere (primarily ARM Ltd and OnLine Media). This has led to a small decline in Acorn, not helped by the main supplier entering receivership and Acorn gmbh closing. This has also led to users losing confidence in Acorn – look at the Tornado project. Would Niall be doing this if the latest superspecial RiscOS4 was just about to appear?

The BBC micro was a powerful machine in it's time. It had competition in the form of Dragons, Speccies and suchlike. The Speccy had 48K compared to the BBC's 32 – but the BBC was so much better in many respects, so much so the BBC chose it for a TV programme (hence the name "BBC micro". Including of course that famous (?) story about Acorn telling Bill Gates that they couldn't possibly take such a retrograde step.

Why then does it appear to all – including die-hard Acorn fanatics – that Acorn is no longer something to admire? Is it the over-hyped need for "industry standard"? "Acorn can't run WordPerfect^{TM} so it's crap"? (it can actually!).

There are many possible theories, the likely answer being combined from the eclectic mess of reasons and excuses therein. I could sit here and type a good theory or even a big thesis to explain why Acorn isn't as good now as it once was, but – like Don Quixote – I'd be more or less fighting enemy windmills.

EPILOGUE:

Don't get me wrong, I'm not anti-Acorn. I've only ever *seriously* owned Acorn computers. Sure, I have/had a Dragon – but for games. I stand behind my BBC micro, my A3000 and my A5000. The BBC and A3000 may be quirky now – but they've given me many years of use. I stand behind my decision to buy Acorn – against the march of MicroSoft^{TM} and people yelling "getta real computer" from all sides – even when a PC user can find a way to reduce my argument to little more than blind devotion. I would mourn the passing of Acorn like America mourned the assassination of J.F.Kennedy. But Acorn is still here isn't it? For how long though? Shapes move outside the window but the dust is beginning to settle here. What we need is somebody to open the door and let the storm in. Maybe that very person is Niall Douglas (Tornado). Maybe not. Whoever you are, we need you. We need you now.

Long live Acorn!

Notice Board

PRODIGY SYSTEMS HAVE GONE SUMMER CRAZY!!!!

CHECK OUT THE REAL-LOW PRICES!

IDE bare-drive (3.5″)	IDE HardCard (A300/A400/A500)
540Mb IDE £155.00	540Mb IDE £205.00
850Mb IDE £200.00	850Mb IDE £250.00
1Gb IDE £250.00	1Gb IDE £300.00

IDE HardCard (A3000/A3010)	IDE bare-drive (2.5″)
265Mb IDE £250.00	265Mb IDE £160.00
540Mb IDE £365.00	540Mb IDE £275.00

SIMMs	CD-ROM
4Mb .. £140.00	IDE 2x .. £175
8Mb .. £250.00	IDE 4x .. £225
16Mb . £400.00	

All SIMMs are fully tested and of the highest quality.
All CD-ROMs come with CDFS and control software.

Backup
420Mb Tape SCSI £200.00
135Mb Syquest IDE ... £240.00
135Mb Syquest SCSI .. £275.00
105Mb ZipDrive SCSI . £240.00

Controlling software comes with the devices,
so no need to wait/write your own.

All prices are *inclusive* of VAT, P&P (UK...), cables, fittings. All interfaces and drives are top quality branded products (Conner/Quantum etc). All drives are formatted for the user. Please ring to ascertain suitability of product before ordering. Please ring for latest prices or other products such as RAM etc.
CD-ROM drives are available in IDE and SCSI forms, please ring for prices.

-- Enquiries	
Tel: 01274 596407.	Address: 15 Red Beck Vale
BBS: Northern ARM (01274) 530831	Shipley
Fidonet: 2:250/372.0	West Yorkshire
EMail:	BD18 3BN
cjackson@digibank.demon.co.uk	

BudgieSoft program listing

AR_Info [1.00] Utility
This utility supports ARMsRace Virtual SysOp door by providing an ARMsRace data file viewer.
NOTE: AR format changed. A new version will be out soon!

CastAVote [2.50] Door; ArcBBS 1.63/64 & ArmBBS
This popular door now comes as a bundled package including VoteEdit (the votes data editor) and VoteConfg (easy-configure CastAVote) and more... Utilities to convert your votes to different formats including CSV.

DoorDocs [5] Textfile
The definitive door writers guide to bits.

HappyHak [1.06] Door; ArcBBS 1.63/64 & ArmBBS
Wind up your users with a faux ‘wardialler’ door.

ArcBBS help [1] Textfile
Addition to !SrcEdit (Acorn’s DDE) to give on-line help on many ArcBBS commands.

InfoSys [1.06] Utility; ArcBBS 1.64 only
INFOrmation SYStem. Use Fidonet for a variety of things that you thought only the Internet could do. Mail batch processing and ftp are all possible.

LastUsers [1.26] Door; ArcBBS 1.63/1.64
Gives a pretty coloured (or mono if you are not an ANSI user) display of the last # users to call your BBS. Includes daily totals count.

Linker [1.00] Door; ArcBBS 1.63/1.64
Links your door to a blockdriver, giving better than 80% throughput in most cases. Could be a link to a modem for follow-on calling, or an internet gate or or or...

ONLINE [1.00] Door; ArcBBS 1.64 & ArmBBS
This parodies the ArcBBS ONLINE command (list users currently on-line). However it does this in real-time so you can watch BBS life as it happens.

Parlez [1.32] Door; ArcBBS 1.64 & ArmBBS
This door provides and BBS inter-messaging facility. You can conduct a private chat or just send a message. Failed messages are uploaded to the user’s mailbox (ArcBBS only).

ReadTasks [2.15] Door; ArcBBS 1.63/64 & ArmBBS
This will display the tasks that you are currently running - not a lot of point, basically a clone of the Archiboard facility - just to see if I could do it. This was BudgieSoft’s first ever door.

ReadUsers [1.00] Utility; ArcBBS 1.64
This utility will scan through the userbase and output information on the users in long form or short form. Useful for taking a ‘snapshot’ of your userbase in case something happens to it. Can be easily converted to ArcBBS 1.63.

RecordUsr [1.00] Door; ArcBBS 1.63/64 & ArmBBS
Records when a user enters a certain menu. Useful for checking the popularity of things.

SetUser0 [1.00] Utility; ArcBBS 1.64
Ever peed-off by the fact that ArcBBS 1.64 won’t allow you to log on as user #0 - the default user? Be peed-off no longer. This utility allows you to nominate another user to be the ‘default’ user. Simply set up this nominated user as you wish the default to be, then run this program... Hey presto - the default user #0 set up as you require. Password/message areas and all.

SysOpChat [1.63beta22] Door; ArcBBS 1.63/64 & ArmBBS
This provides a nice split-screen SysOp chat. Features: Macros, spooling, snapshot, easy-to-configure, chat ‘available’ times, automatic excuse, choice of excuses, configurable screen displays, spool facility for ANSI and data and 1stWord+ as well as ASCII (can display in SysOp’s chat window or full-screen. Can even display a relocatable module!), international support, unique bleeping so you know you are being paged.

TopFeatur [1.00] Utility; ArcBBS 1.63/64
Like a glorified RecordUsr, with graph display. Definitely techie-fodder this. Can be converted to ArmBBS or whatever.

TTXDoor [1.00alpha] Door; ArcBBS 1.63/64 & ArmBBS
On-line teletext door - converts teletext to ANSI. :-)

ViewFile [1.00beta] Door; ArcBBS 1.63/64 & ArmBBS
Upon DaviD Dade’s suggestion, I am working on a door to allow the SysOp to view certain files, like the BBS logs.

BudgieSoft...
>3Mb of hand-created code and still going strong. RSI exists!:-)

BORED?
Call ArcTic BBS on:
+44 181 903 1309 (28800bps)
+44 181 903 1308 (14400bps)
SysOp: Steve Pursey