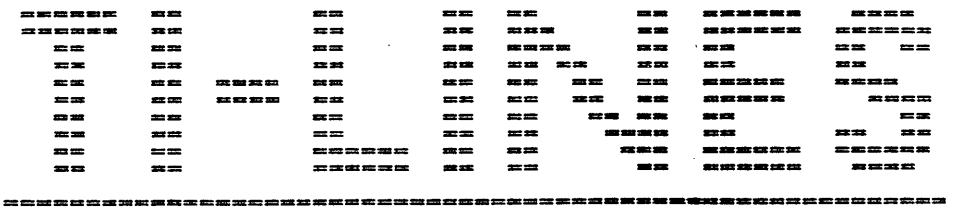


Volume 2, Issue 10 March 1st., 1986

TI-LINES is produced and published by:

Peter G. Q. Brooks

96 Banbury Road
OXFORD
OX2 6JT

Tel: Oxford 50822

Tel. (02357) 2360
WESSEX PRESS
9-11 Mill Street, Wantage, Oxon OX12 9AB.

Oxon residents are offered a subsidised subscription at £1.56 p.a., payable either as 12 second class postage stamps or by cheque.

THIS OFFER APPLIES UNTIL MAY 1986 ONLY

TI-LINES is available on Associate subscription to Users resident outside Oxfordshire, for £10 p.a. Back issues are £2 including post and packing. New subscriptions begin with Issue 1 of the current volume, up to and including the current issue, regardless of the number of issues elapsed.

Contributions should be submitted either on diskette in TI-Writer compatible files (DIS/VAR 80 or DIS/FIX 80), or in a form which is as legible as possible. Artwork should fit within an A4 area and should not contain colour. Very high contrast line drawings are preferred and these may be produced by arrangement with the publisher.

WARNING! ERROR IN DIAGRAM FOR 32K RAM

Index to articles

O X O N T I U S E R S N O W N U M B E R 1 2 6

EDITORIAL

FOR APOLOGY=1 TO INFINITY::PRINT "OOPS, OH DEAR"::NEXT APOLOGY

The observant reader may have noticed something amiss with the February issue of TI-LINES. No, I don't mean the fact that it actually came out while the month was still called February. I mean the fact that it had an odd un-numbered sheet stapled at its centre. Without any indication of what it should refer to, just to add a bit of spice.

Having rushed through that issue overnight, as with the January issue, in order to try and catch up with myself, I found after having neatly numbered every page that I had missed out two A4 pages of diagrams from GRAHAM WOLSTENHOLME's article. I swiftly stuck them in without adding any numbers, which wasn't too drastic, except that I forgot to label them fully.

So now you know. You may not know why, but you know how. Lack of brain.

TOTALLY UNSOLICITED TESTIMONIAL

I write in praise of the excellent newsletter produced by the TI NOVA SCOTIA group, TINS, from which I have often selected material in the past, and from which I hope in the not too distant future to select the source code from the RS232 card (always assuming that permission to reproduce can be obtained from the original author).

My regard for the TINS newsletter has absolutely nothing whatever to do with the fact that my articles on Enhanced BASIC plus one or two others are being serialised in that worthy publication. The layout looks a lot better than it did when I published them, too.

PERIPHERAL PROBLEMS

OTIUser DAVE CARR wrote to me in January outlining a peculiar problem that he has been experiencing with his PEB. Initial powering up is OK both with and without the PEB.

After a period of time, probably about 30 minutes to an hour, if he changes from EDITing to RUNning or SAVEing or something similar, everything locks up. If he uses the reset switch everything clears, but without executing the usual power up routine. The screen goes blank as it does with Extended BASIC when it searches a drive for the LOAD file.

If the PEB and disk drives are shut down, then the minimal system powers up normally.

Leaving the PEB switched off for a while seems to enable it to catch its breath, but when it has been reconnected a while, history repeats itself.

Dave has dismantled his system, cleaned all sockets and connectors, and then put it back together again. To no avail, the problem is still there, and he is no nearer a solution.

If anyone has any bright ideas, or has been experiencing similar problems, perhaps they would like to contact me and I will pass your information on as well as publishing anything helpful in TI-LINES.

~~~~~~

BASIC INCOMPATIBILITY

I am advised that the MATCHBOX RAM EXPANSION, details of which were given in the last issue, might not co-exist with the latest EXTENDED BASIC II PLUS. Apparently both make demands on the console's power supply, and it may not have the necessary extra capacity to power them both. If anyone has any more details perhaps they would let me know so that I can advise members one way or the other.

~~~~~~

LAMPOONIST LAMPOONED

ANN HON, who is probably none other than the devilish ANN ONYMOUS-SCRIBBLER, sent me a piece of verse and a cartoon, presumably dumped out on her printer when she had plenty better to do:

There was a young man named Pete Brooks

Who babbled about writing books

He won't go to bed,

'Cos the hair on his head

Might rub off and spoil his good looks!

This just goes to show that Ann ought to be registered as partially-sighted. I see Pete Brooks most mornings, and young is not an apt description. Not only that, but he had sideboards (sideburns to some of our readers) last time I saw him, although the ratio of bald to non-bald is just about right.

~~~~~~

YOU HEARD IT HERE FIRST...

Anyone who watched the TV programme MICROLIVE recently might have had a vague feeling of deja vu (it's alright, I did French 'A' level and it helps me keep my hand in) when listening to the bit about Racter and Eliza. That is because Racter's exploits (The Policeman's Beard Is Half Constructed) were reported in TI-LINES a while back. Also in the programme was the original for one of my favourite anecdotes about Speech Recognition, with which I will regale you just in case you missed the programme.

The demonstration covered a flight information system operating in an American airport. You ring up the airline and its computer does the work,
~~~~~~

listening to your replies to simple questions and analysing what you say. The demonstration on the phone went something like this:

Computer: From which city do you wish to fly ?

Customer: Los Angeles.

Computer: Los Angeles. To which city do you wish to fly ?

Customer: Newark.

Computer: Newark. Flights from Los Angeles to Newark. On which day of the week do you wish to fly ?

At this point the "customer" coughed into the phone (Hitchiker's fans will know that this is dangerous!):

Computer: (Pause...) Saturday.

Everyone fell about laughing.

Other things cropped up in Microlive's programme (not program, notice), one of which was of particular interest to me since it provided supporting evidence for one of My Pet Theories. This Pet Theory is that folks in the future will spend more and more of their time pursuing what we would call Further Education. The Age Ahead will probably be based less on the need to work, and more on the urge to discover, to learn.

In France, there is something called the University of the Third Age. Such Universities are catching on, and there is now at least one in the UK. What is the Third Age ? It's the age you reach when you retire... A sort of Open University for the elderly, it provides not only a focus of social interaction, but also what is now recognised as necessary: a way of keeping the mind active. The general view, for example, of the physical state of the group of people known as Chelsea Pensioners, is that the group has a very low incidence of senile dementia because they keep themselves very active intellectually. Many people believe that you cannot teach an old dog new tricks, but the truth is that extremely mature students are better able to cope with the requirements of study than are the young. This probably has more to do with attitudes instilled at a young age (people retiring now learned their school habits in the Thirties, when self discipline had a much greater priority than it has now), but either way, the French have shown that education is something to be pursued even in your eighties.

~~~~~~

32K RAM EXPANSION: WARNING

BRIAN ABRAMS and DAVE HEWITT have notified me that there may be a serious flaw in the diagram published in the last issue of TI-LINES. Figure 1 on the INSERT shows the chips used and their links; it is probable that the links between pins 14 and 15 on each chip should not be there at all, and damage to the computer could result. I have notified CLIVE SCALLY of TI-EXCHANGE, who tells me that he has given out 100 copies of the details provided by GRAHAM WOLSTENHOLME without any complaint, and many people have successfully completed the project.
~~~~~~

I have contacted Graham for clarification, and he is looking at the material published in TI-LINES and will report to me shortly (probably too late to get in this issue unless I delay it).

In the meantime, if you intend building the 32K expansion, hold your horses for the present until I have the situation clarified.

~~~~~~

32K RAM EXPANSION: MORE SUPPLIERS

BRIAN ABRAMS (see above) also notified me of two further suppliers of the chip used in the 32K project originated by PHIL WEST and BERNIE ELSNER of Australia. The first is STC, on 0279 26777, who charge £3.27 per chip, with an overall post/packing charge of £3.15. You have to allow two weeks for delivery.

WATFORD ELECTRONICS on the other hand, charge £3.50 per chip, or £6.70 for a pair, with only £1 post/packing. These are available ex-stock. Contact Watford Electronics on either 0923 40588 or 0923 37774. Thanks, Brian.

~~~~~~

THE ONGOING HELLO SITUATION...

OTIU is still growing in fits and starts (hope I got that spelled the right way round!) and this month sees us welcoming DENNIS TRICKER, PETER STIMPSON, GERALD COLLINS, R. F. JACKSON, PETER HILDEBRAND, PETER BERRY, COLIN GAUNT, RICHARD IRELAND, W. REED, and Mrs J. PATTISON. Some of our most recent subscribers have asked for attention to be paid to the use of MODEMs, and if you are looking for a modem why not give GORDON PITT a ring on 0922 476373 and try to help boost the numbers which he needs to secure a special price for a bulk order ?

~~~~~~

DELIVER THE (HALF-BAKED) CAKE ?

OTIUser DARYL DAFFIN contacted me recently over a program from the new TSC Catalogue - DELIVER THE CAKE, code GA0003 - which has some erring lines. It looks as though the program was edited at some time, and a few lines were removed. I suspect that these may have been CALL SOUND() statements, but I cannot be sure. The result is a GOSUB to a line which does not exist, but because of the way the program has been modified, does not seem to cause any immediate problem (e.g., such as a crash) although it does cause some vital pieces of information to be omitted (lines 3520 onwards). The missing lines are 3550 to 3570, and there may also have been some editing of lines 3730 and 3740 (because they also appear to be missing.). If anyone can shed some light on the matter, feel free to shed away.

~~~~~~

NOTES FOR YOUR DIARY

The next TWO TI-EXCHANGE meetings have been confirmed by CLIVE SCALLY. The first is to take place in LEEDS (venue not yet known but give me time) on SATURDAY MAY 3rd., while the second is scheduled for SEPTEMBER 25th in BOURNEMOUTH.

Clive and Audrey will also be manning part of the ACC stand at the PERSONAL COMPUTER WORLD SHOW in September, and Clive would welcome volunteers to keep the presence going during the exhibition.

Further details will be available on 0273 503968 after 7 pm., and I plan to be at all three venues unless something pretty drastic stops me!

~~~~~~

TI-WRITER MANUAL CONFIRMATION

OTIUser BILL MORAN has kindly provided hard evidence that TI are definitely selling their TI-Writer manual for £3.50 - complete with folder - in the form of the packing note from his recently-bought manual. If anyone tries to enquire from TI about the availability of this manual, and is told that no such sale is possible (as has happened to at least one member!) tell them (politely of course) that they're wrong! The signature on this particular packing note is G or J BAPTISTE, if you want a name to bandy about.

~~~~~~

MODERN OBSOLESCENCE

I recently watched two imported American TV programmes, both of which made use of robots or computerised speech. In both cases the voices were totally devoid of expression, consisting of a monotone without inflexion, rather like the early Fifties films which used such techniques to indicate to the audience exactly who was the alien/robot and who wasn't. The trouble is, both of the TV programmes were modern, so they didn't have any excuse. I wonder when the film/TV producers are going to wake up to the fact that dull, lifeless, mechanised speech went out with the Ark ? Maybe someone ought to send the studios a tape of the TI singing or something, just to put them in the picture, if you'll forgive the pun.

~~~~~~

RECOURSE TO REC CLAUSE

While reading the NORTHWEST OHIO 99'ER NEWS, the newsletter of the OH-MI-TI and NEW HORIZONS Users Groups, I picked up on several items which will be of interest to some OTIUsers, and one thing in particular which caught my eye was a brief presentation by STEVE PATTERSON of the New Horizons group, entitled ADVENTURES IN BASIC - #2.

In it, Steve provided a console BASIC program to delete all the files from a given disk ("sweeping" the disk, as it is known). If you have an expanded system, the excellent DM1000 will do this for you in a trice, but for those with a system which is a step up from the minimal system (console and cassette leads), Steve's program is a useful alternative to the tedious process of having to either initialise the disk again, or manually delete all the files from either the Immediate mode or using Disk Manager.

However, Steve noted that his program seemed to need at least four passes in order to do the job fully, although he was not sure why it did not do a full sweep in just one pass.
~~~~~~

I wondered too, so I spent some time working on the problem, and I believe I may have the solution. I began by writing a short program to create a number of "empty" files on a disk so that I had something to delete (i.e., not the files for this issue of TI-LINES!). It goes like this:

```
100  CALL CLEAR
110  FOR I=1 TO 30
120  A$="DSK1."&SEG$(STR$(I/1E3)&"00",2,3)
130  PRINT A$
140  OPEN #1:A$, OUTPUT, DISPLAY, VARIABLE 80
150  CLOSE #1
160  NEXT I
```

Line 120 simply creates a file name which is a number between 1 and 30, but which contains leading and trailing zeros so that everything appears in numerical order in the directory.

Now, if you have decided to try this out and have entered and run the above routine using a blank, initialised disk, if you use Disk Manager to look at the catalogue you should see 30 files beginning "001" and going through to "030".

Go back into console BASIC and key in this routine:

```
100  OPEN #1:"DSK1.", INPUT, INTERNAL, RELATIVE
110  INPUT #1:A$
120  IF A$="" THEN 160
130  PRINT A$
140  DELETE "DSK1."&A$
150  GOTO 110
160  CLOSE #1
```

What you are doing here is to open a file to the Directory, reading in the file names one by one, and deleting them. Or not, as the case may be.

If you run the program once, using the previously-created files as your cannon-fodder, you may notice that the deleting process seems to skip every other file, and once one pass has been completed, if you examine the disk directory using Disk Manager, ..er.. there are still at least half of the files left on the directory.

So what has gone wrong ?

It is a little obscure, but the culprit is the file format. The directory is a RELATIVE file, which means that when reading or writing items to that type of file, the computer keeps an internal note of exactly where it has got to (rather like the internal record kept when READing DATA).

What appears to happen is this: the first entry is read in using INPUT #. That first entry is actually the disk name itself, which cannot be DELETEd. The computer moves on to the next entry, which is the first "real" entry in the directory. Accordingly the computer deletes that file, and as a result what was previously the SECOND entry in the file now becomes the FIRST entry (as the original first entry has just been deleted). However, because the computer is using a RELATIVE file, it has just updated its internal pointer to show itself where it is, and this pointer now indicates that it should be looking at the second entry in the file. Are you with me ? What is now the

second entry was, before the first deletion, the THIRD entry. Because of this internal updating, the previous second entry, now the FIRST, slips through the net, so to speak.

This "escape" from the deletion process continues for every other entry in the file, until the above program encounters a null entry which signals the end of the directory.

Thus after the first pass, only half the directory has been deleted, and if you were to run the program again, the same effect would occur, deleting only half of the entries again. This is why Steve's program needs more than one pass to do its job.

Salvation lies in the REC clause. If you include REC() in the INPUT # statement, the computer is commanded to change its internal pointer to what ever value REC specifies.

Now, as the deletion of an entry from the directory causes each remaining entry to "move up one", as it where, we need to use the REC clause to keep changing the internal pointer each time so that it ALWAYS points to the FIRST entry in the directory.

REC(0) is the disk name, so we don't want to get stuck trying to delete something which cannot be deleted! The solution is REC(1), and the correct statement in line 110 is:

```
110  INPUT #1, REC(1):A$
```

That's all it takes. Use REC(1) and ALL the existing entries will be deleted in turn.

I have to say though that it can be such a slow business, you might be better off using Disk Manager to re-initialise the entire disk...

There is another way in which the program can delete all the files and NOT need the REC() clause. Can anyone see how ? Yes, I know, I just want to see if YOU know...

~~~~~~

C O N T A C T S

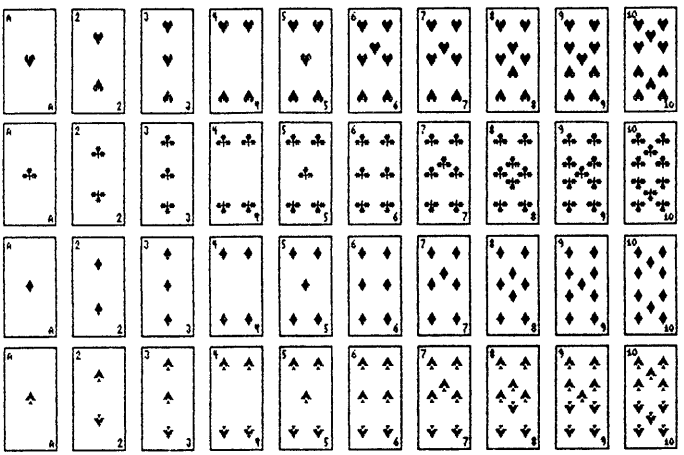
GERALD COLLINS — NEWLANDS, 6 NEWLANDS ROAD, BOSCOMBE EAST, BOURNEMOUTH, DORSET BH7 6NX
PHONE: 0202 433704

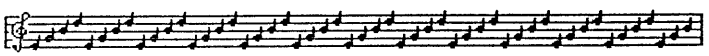
PETER BERRY — 31 CLAY BUTTS, GRIMESCAR VALLEY, HUDDERSFIELD, W. YORKS HD2 2FW

NICHOLAS FROST — 7 ST. MARYS ROAD, FAVERSHAM, KENT, ME13 8EU
PHONE: 0795 533223

~~~~~~

TI WRITER GRAPHICS





BY JAMES STRINGFELLOW

THESE GRAPHICS WERE PRINTED DIRECTLY FROM THE KEYBOARD
USING TI WRITER AND EPSON MX80 PRINTER

Graphics can be included in your letters printed by TI Writer, directly from the keyboard. This can be useful to personalize your letterheads. The printer manual explains how to do graphics.
A Basic program to print this ■ would be as follows:

```
10 OPEN #1:"RS232.BA=4800"20 PRINT #1:CHR$(27);CHR$(75);CHR$(6);CHR$(0)
;CHR$(63);CHR$(63);CHR$(63);(63);CHR$(63);CHR$(63);CHR$(63) :: CLOSE #1
```

A dot matrix printer prints 7 dots vertically each time it receives a ASCII code from your program. See ASCII character code chart.

With TI Writer, it is only for numbers 0 to 31 that you will use CTRL U. I have enclosed a chart giving the equivalent numbers for each character that you will see on the screen; also a grid that will help you design your graphics.

To print this ■ you would see on the screen 'ĸK??????
broken down as follows for a better understanding:

'ĸK??
↑ This is CHR$(27) followed by K to turn on graphic mode.

'ĸK??
↑ This position indicates the amount of graphic lines following, up to 127, in this case "six".

'ĸK??
↑ This position can be from 0 to 4, each unit equals 256 vertical dot lines i.e 1=256, 2=512, 3=768, 4=1024. In this case "zero".

?????? are the number of vertical dots to turn on, six times 63.

Resume:
- The 'ĸK turns on graphic mode.
- The small ? indicates there will be 6 data following.
- The small ? in the position of bigger graphic data.

```
Dot 7 = 64
Dot 6 = 32      Examples:
Dot 5 = 16
Dot 3 =  8      Dots 64+1      = 65 or "A" would print :
Dot 3 =  4      Dots 64+8+1    = 73 or "I" would print :
Dot 1 =  2      Dots 1+2+8+16  = 27 or "ĸ" would print :
Dot 1 =  1
       ---
Total  127 .... giving you this graphic  !
       ===
```

In order to print a data, add the value of the dot positions together.

127 can be used by FCTN & V but you will see nothing on the screen.

→

To draw a straight line like this

---

type in the following lines using CTRL U to see this on the screen

'bKò1
ìì
ìì
ìì

'bK This turns on graphic mode.
ò1 These two numbers give the amount of graphic dot lines to print.
The first number can be from 1 to 127.

'bK~ò
↑ The ~ here indicates 126 vertical dot lines.

'bKò1
↑ The 1 here indicates 256 vertical dot lines.
This number can be multiplied by 256, i.e. 1 in this position = 256.
When using more than 80 graphic lines you must add a carriage return to your printer, i.e. "RS232BA=4800.CR".

To print a ♡ this is what you would see on the screen.

'bK9òOHD"'"DHO

To print a ↑ this is what you would see on the screen.

'bK9òA'c89s98tA

'bK9òA'c89s98tA

The letter K turns on double graphic printing.
The small 9 is the amount of graphics to print.
The ò is not used because the amount of lines is under 128.
The nine following codes are the vertical graphic dot lines.

I hope that these examples will enable you to make your own graphics with TI Writer.
I will be happy to answer any questions that you may have.

James Stringfellow
23 rue Pasteur
78700 CONFLANS
FRANCE

→

CONTROL CODES

USING CTRL U

0	ø	Shift	@	32		64	@	96	`
1	¡	Shift	A	33	!	65	A	97	a
2	²	Shift	B	34	"	66	B	98	b
3	³	Shift	C	35	#	67	C	99	c
4	´	Shift	D	36	$	68	D	100	d
5	µ	Shift	E	37	%	69	E	101	e
6	¶	Shift	F	38	&	70	F	102	f
7	·	Shift	G	39	'	71	G	103	g
8	¸	Shift	H	40	(	72	H	104	h
9	¹	Shift	I	41	)	73	I	105	i
10	º	Shift	J	42	*	74	J	106	j
11	»	Shift	K	43	+	75	K	107	k
12	¼	Shift	L	44	,	76	L	108	l
13	½	Shift	M	45	-	77	M	109	m
14	¾	Shift	N	46	.	78	N	110	n
15	¿	Shift	O	47	/	79	O	111	o
16	À	Shift	P	48	0	80	P	112	p
17	Á	Shift	Q	49	1	81	Q	113	q
18	Â	Shift	R	50	2	82	R	114	r
19	Ã	Shift	S	51	3	83	S	115	s
20	Ä	Shift	T	52	4	84	T	116	t
21	Å	Shift	U	53	5	85	U	117	u
22	Æ	Shift	V	54	6	86	V	118	v
23	Ç	Shift	W	55	7	87	W	119	w
24	È	Shift	X	56	8	88	X	120	x
25	É	Shift	Y	57	9	89	Y	121	y
26	Ê	Shift	Z	58	:	90	Z	122	z
27	Ë	FCTN	R	59	;	91	[	123	{
28	Ì	FCTN	Z	60	<	92	\	124	¦
29	Í	FCTN	T	61	=	93	]	125	}
30	Î	Shift	6	62	>	94	^	126	~
31	Ï	FCTN	U	63	?	95	_	127	DEL

64											
32											
16											
8											
4											
2											
1											

SORTING AND SEARCHING

Peter Brooks March

PREAMBLE

First, a few notes about the simple sorting routine published in V2.6. A couple of readers rang me to ask how to expand it - they'd encountered the lack of "dynamic" allocation of space for arrays. This jargon simply means that if you intend having more than 11 entries in your list to be sorted (elements 0 to 10 inclusive), you have to tell the computer to reserve more space first. The DIM command (short for DIMENSION) allows you to set the maximum size for the list you are going to use. To create space for 100 entries, use DIM variable(99) if you intend counting from zero, or DIM variable(100) if counting from one. You can of course count from anywhere you like - from 17 if you wish - as long as you DIM enough space!

Just in case there is any misunderstanding, where I have used the word "variable" in "DIM variable()" you would of course use whatever variable name you choose.

In order to sort 100 items, counting the list from 0 to 99 entries, use DIM variable(99). Other changes need to be made. All loops have to be adjusted to take account of the increased list size, so the input, sort, and output loops all need to be changed.

Referring to the listing in V2.6, lines 120, 160, and 250 need to have the 9 changed to 99. (Don't forget to also insert a DIM line at the beginning of the listing - say at line 105 - or you'll wonder why nothing works!).

There is one problem which you will no doubt encounter: the larger the list of data to be entered, the longer it takes to do so and the more likely you are to make mistakes. If you have large amounts of data it might be better to look at alternative "input" techniques.

The simplest is to place all your data in DATA statements and use READ instead of INPUT in line 130. This is not practical for any really large list, but can be helpful when testing different sorting techniques for efficiency for example.

Another alternative is to place the data on an "external storage medium" - record it on tape or disk (or MiniMem, or Expmem). As far as tape is concerned, you are rather limited, both by the small range of formats and by the poor reliability. Disk storage is a different matter, and it can be very useful to set up a file using a data format equivalent to DISPLAY VARIABLE 80, the format usable by TI-Writer and the Editor of the Editor/Assembler. Don't forget that these days you don't HAVE to have the TI-Writer module, not if you've got Extended BASIC and an expanded system, as the Enhancements (99% of TI-Writer) are Public Domain and there are some excellent machine code loaders about (e.g. FunlWriter).

Anyway, enough of that (unless YOU want more). Back to the main theme.

This issue looks at turning our Bubble sort into a Tag sort. We took a preliminary look at the principle of Tag sorting back in issue V2.3, where

two lists were used. One contained forenames, the other contained corresponding surnames. There can be a number of such "linked" lists, all needing to be sorted in parallel with the "key" (see V2.3 to refresh your memory).

An example might be a list of members of a group: their titles and forenames or initials, their surnames, their addresses (arranged in individual lists, containing the street number, name, local town, postal town, county, and post code, so that sorting could be performed using any one of the address items), the subscriptions they pay, their status (associate, honorary, etc.) and so on. You can probably think of better examples so I won't labour the point.

THIS ARTICLE REALLY BEGINS HERE...

For the example to be used in this article, I will use three lists:

1) Surname

2) Postal town

3) Membership status

To keep things simple I will use three string arrays:

S$() for Surname

T$() for Postal town

M$() for Membership status

This is perhaps too simple - there are all sorts of pitfalls which could crop up - but suspend your queries about what happens if ten people with the same surname live in the same postal town, have the same membership status, but are not related.

We'll have 20 surnames in our first list, S$(), with 20 corresponding postal towns - T$() - and status' - M$(). A surname in list S$() at position 'n' will have its corresponding postal town in position 'n' in the T$() list, and the membership status in position 'n' in the M$() list.

Note that we could have used ONE array for the lists - a TWO DIMENSIONAL array - i.e.:

```
DIM L$(2,19)
```

Remember that the elements are numbered 0 to 19, giving 20 elements, by 0 to 2, giving 3 elements.

Thus, L$(0,0 to 19) would hold the surname list

L$(1,0 to 19) would hold the postal town list

L$(2,0 to 19) would hold the membership status list

In terms of programming, the Tag sort swap routine would benefit from such a format, as you will hopefully see, since a simple FOR-NEXT loop could take care of the whole swap procedure.

Back to the 3 array/list example.

You might have wondered about possible methods of filling these arrays with the information which we will be using. There are several approaches:

a) Manual entry at the keyboard

b) Embedding in the program listing as DATA statements

c) Embedding in the program listing as a series of LET assignments

d) Transfer from cassette, disk, or other file (e.g. MiniMemory)

e) Pseudo-random generation from basic components by a routine within the program

Generally, sorting procedures are usually applied to data obtained by method (d).

In this example, to allow us to rerun the example programs (and subsequent programs later in the series), we will embed the information in DATA statements. If you never quite managed to fully grasp the function of READ, RESTORE, and DATA, read on.

Here are the lists we will be using:

Position	S$()	T$()	M$()
0	BROOKS	OXFORD	P
1	BROOME	LINCOLN	A
2	FIELD	READING	A
3	COOPER	OXFORD	A
4	BRUGGE	RHEIMS	O
5	HERRESTHAL	HAMBURG	O
6	PAULI	CAPETOWN	O
7	CRUMB	NOTTINGHAM	A
8	FUTTOCK	CREDITON	A
9	MANTLE	MORDEN	A
10	JAMIESON	WINDSOR	A
11	ROLANDS	BIRMINGHAM	A
12	BURNETT	LIVERPOOL	A
13	MASSIF	PARIS	O
14	MARCHAM	DONCASTER	A
15	LEWIS	HULL	A
16	GARDNER	MILWAUKEE	O
17	MATHIS	LAS VEGAS	O
18	TRINDER	CANBERRA	O
19	SALMON	MONTREAL	O

The membership status list has A for Associate, O for Overseas, and P for Pillock...no, Perpetual.

The postal towns may not be accurate, but then this is not meant to be a geography lesson.

Continuing in the make-life-easy vein, all entries will be in upper case (capitals). This should prevent awkward situations where "LIVERPOOL" and "Liverpool" would be seen by the computer as totally different locations (as would "LiVeRpOoL"!).

Now for the program, and the use of DATA statements.

To make it relatively easy to locate a particular entry, each DATA statement will contain one entry from each list. This has the added advantage of allowing a single READ instruction to assign an entire entry to each list.

In addition, the DATA statements will be assigned line numbers which are related to their initial position in the lists: entry 0 (BROOKS) will be in the DATA statement at line 1000, while entry 19 (SALMON) will be at 1019.

This is really to help you debug the program listing should you experience difficulty (due, hopefully, to your bad typing and not mine!). There is a series coming up on Debugging: How She Is Spoke, so I won't go into great detail - if you have problems and you can't solve them, then drop me a line.

We will use a structure for the program similar to that used for the listing given in V2.6, so you should be able to follow it. The program will consist of three main processing sections and one DATA section. The three are:

(a) Data assignment (equivalent to INPUT)

(b) Sort routine

(c) Output of sorted lists

When we look at the sorting routines you will begin to appreciate why the use of a single two-dimensional array is preferable instead to the three one-dimensional arrays.

DATA ASSIGNMENT

This is reasonably straightforward. This section consists of a loop from 0 to 19 within which there is a READ. If you've never quite grasped READ before, here's your chance.

READ is used in conjunction with two other BASIC words: RESTORE and DATA. You may have seen program listings in which there were lines beginning with the word DATA and followed by a list of things separated by commas.

The comma, as is often the case in TI BASIC, acts as a "separator", helping the computer to distinguish in a list between the end of one item and the beginning of another. DATA statements are a little like REMs - the computer doesn't try to execute the contents of DATA statements either.

The difference arises when READ is used. The first time that the computer encounters a READ while running a program, it starts looking from the beginning of the listing for a DATA line. When it finds one, it notes the location and stores the information internally. The READ instruction is always followed by one or more variables; READ A or READ C,Z$,Y(N) and so

on. Having found the first occurrence of a DATA statement, the computer proceeds to copy the items in the data list into the variables in the READ instruction. If there is one variable, the computer copies one item. If there are five variables, the computer copies five items. Naturally the items and the variables must match - you couldn't assign a string like "DOG" to a numeric variable, for example, although, confusingly, you can assign a number to a string variable!

The computer keeps track internally of how far into the DATA list it has got - if it reaches the end of one DATA statement it will look for the next in the listing, and if it has only got partway through a DATA statement it will "remember" exactly where it is, ready for any further READ instruction.

Sometimes it may be necessary for a list of DATA to be read more than once (perhaps when setting up a game board for a fresh game) and this is where RESTORE comes in. On its own, RESTORE causes the computer to change its internal record of where it is in the maze of DATA items. The computer goes back to the first DATA statement in the listing, ready to begin again.

If RESTORE is used with a line number, e.g. RESTORE 5020, the computer will reset its internal marker so that it is ready to continue READing from the DATA statement in the specified line number (in the example, line 5020).

All clear ? Good. Be quiet at the back.

In the case of the three lists to be used for sorting here, the DATA is organised so that each statement consists of three items - surname, postal town, and status. The READ will therefore read all three into their respective arrays at one go: READ S$(), T$(), M$(). The value within the brackets in each case will be given by the variable used to control the FOR-NEXT loop:

```
FOR P = 0 TO 19
READ S$(P),T$(P),M$(P)
NEXT P
```

To save space you can of course cram as many items into a DATA statement as possible; I have (hopefully) designed the DATA layout so that errors will be easier to spot and correct.

THE SORTING ROUTINE

Basically the procedure here is the same as in V2.6. There are a couple of differences. The most obvious is the fact that we are dealing not with one array, but with three. This entails an expanded swap routine, catering for all three. Less obvious is the fact that because we have three arrays or lists, there are three different possible "keys". I have chosen to use the surname (S$()) as the key here, which means that the S$() array will be sorted and the T$() and M$() arrays will receive the same manipulation as is applied to S$(), except that only S$() will tested in the IF..THEN section. If you want to sort using another key, either T$() or M$(), you simply need to change both occurrences of S$ in the IF...THEN test to T$ or M$ - it is as simple as that. The actual swap sequence can stay the same.

The use of a single, but two-dimensional, array removes the need for a large complex swap routine; however, the discussion of the programming involved will turn this into an article on programming techniques rather than on

Sorting, so I won't go into further detail here (I'll cover it next time instead!).

The test section involving S$() is basically the same as that in line 170 in V2.6, although the destination line number is different because the swap routine is that much larger.

THE SWAP ROUTINE

To save using more variables than is absolutely necessary for clarity, I have used G$ to temporarily store any item which is being swapped.

For S$():

```
G$ = S$(P)
S$(P) = S$(P-1)
S$(P-1) = G$
```

For T$():

```
G$ = T$(P)
T$(P) = T$(P-1)
T$(P-1) = G$
```

For M$():

```
G$ = M$(P)
M$(P) = M$(P-1)
M$(P-1) = G$
```

The swap flag is set as before: F = 1, and don't forget that the loop will now be FOR P = 19 TO 1 STEP -1.

THE OUTPUT ROUTINE

Almost identical to that in V2.6, here not only is the loop run from 0 to 19 (in place of 0 to 9), but the PRINT statement has been altered to take into account the increased information. There are four items to be printed: the current value of the loop control variable (P); the surname; the postal town; and the status. To tidy up the printout I have used TAB() to create a "formatted" output. That is to say that each of the four items will be printed starting at a specific column on the screen. The subject of print formatting is intended to be covered in a future article, so I won't go into detail here.

The new PRINT statement is:

```
PRINT STR$(P);TAB(4);S$(P);TAB(15);T$(P);TAB(28);M$(P)
```

Here is the full program listing (take care over the DATA section from line 1000 onwards):

```
100 CALL CLEAR
110 CALL SCREEN(8)
120 DIM S$(19),T$(19),M$(19)
130 FOR P=0 TO 19
140 READ S$(P),T$(P),M$(P)
150 NEXT P
160 F=0
170 FOR P=19 TO 1 STEP -1
180 IF S$(P))=S$(P-1)THEN 290
190 G$=S$(P)
200 S$(P)=S$(P-1)
210 S$(P-1)=G$
220 G$=T$(P)
230 T$(P)=T$(P-1)
240 T$(P-1)=G$
250 G$=M$(P)
260 M$(P)=M$(P-1)
270 M$(P-1)=G$
280 F=1
290 NEXT P
300 IF F=1 THEN 160
310 CALL CLEAR
320 FOR P=0 TO 19
330 PRINT STR$(P);TAB(4);S$(P);TAB(15);T$(P);TAB(28);M$(P)
340 NEXT P
350 END
1000 DATA BROOKS,OXFORD,P
1001 DATA BROOME,LINCOLN,A
1002 DATA FIELD,READING,A
1003 DATA COOPER,OXFORD,A
1004 DATA BRUGGE,RHEIMS,O
1005 DATA HERRESTHAL,HAMBURG,O
1006 DATA PAULI,CAPETOWN,O
1007 DATA CRUMB,NOTTINGHAM,A
1008 DATA FUTTOCK,CREDITON,A
1009 DATA MANTLE,MORDEN,A
1010 DATA JAMIESON,WINDSOR,A
1011 DATA ROLANDS,BIRMINGHAM,A
1012 DATA BURNETT,LIVERPOOL,A
1013 DATA MASSIF,PARIS,O
1014 DATA MARCHAM,DONCASTER,A
1015 DATA LEWIS,HULL,A
1016 DATA GARDNER,MILWAUKEE,O
1017 DATA MATHIS,LAS VEGAS,O
1018 DATA TRINDER,CANBERRA,O
1019 DATA SALMON,MONTREAL,O
```

Don't think that TAG SORTING is confined to Bubble sorts - it is a variation which can be applied to all sorting techniques.

There is one thought, though. In the example here, the original sequence or arrangement of the lists is lost when the sort takes place (forgetting for the moment that the lists are held in DATA statements!). Other sorts which we will examine later use a separate array to hold the sorted results, but it would be possible to store the original location of any key with the key itself. I wonder if anyone can see how ? (I wonder if anyone cares!).

BULLETIN BOARD

SALE FOR SALE FOR SALE FOR SALE FOR SALE FOR SALE FOR SALE FOR SALE FOR SALE

oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo

OTIUser PETER CALCRAFT is putting his MYARC 128K card up for sale. It is only a few months old, and Peter's reason for selling is that it is incompatible with his Pascal system. It cost £230 and he is asking £180 (he will cover postage costs if you cannot collect it yourself). There was a review of the card in TI-LINES V2.2 by HOWARD GREENBERG if you want some background, or you can get all the details from Peter direct, either by phone: 0305 67658, or at 13 Royal Mews, Princes Street, DORCHESTER, Dorset, DT1 1RL. I have used one of these cards before, and they can make life very easy when using things like TI-Writer, for example.

oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo

DAVE HEWITT is still producing his home-grown PIO interface. It is available boxed for £55, unboxed for £45, or as the pcb alone at £10, and the EPROM at also £10 (circuit diagrams provided) if you wish to build it yourself. If you drop Dave a stamped, self-addressed envelope he will send you a sheet of information describing the PIO in more detail. If you purchase a unit from him, he will also throw in a free copy of his own WordWriter word processing program.

You can contact Dave on OXFORD (0865) 863565 or by writing to him:

DAVE HEWITT
5A Lower Whitley Road
Farmoor
OXFORD
OX2 9NU

(see CLOSEFILE for an offer you might not want to refuse!)

oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo

Over the page you will find a list of the modules/disks/cassettes which can be provided through me. The availability may change, and if you order an item which has been sold we will still attempt to obtain it for you; if we have to admit failure, you will of course be reimbursed. Post and packing are included, and I understand that all orders will be despatched using the Compensation Fee Parcel service so that any losses can be recovered.

Contact me if you are interested, orders will be filled on a first-come, first served basis. In a few cases there is only one of each item, in most of the others there are several. A few books are available too, although I am unsure as to exactly how many there are. The best advice is to put your order in as quickly as possible!

~~~~~~
~~~~~~

TITLE	££.pp	FORMAT
ABM CONTROL/CAVERN HUNT	2.50	CASSETTE
ABM CONTROL/FROGLET	2.50	CASSETTE
ADD & SUBTRACT 1	5.00	CARTRIDGE
ADD & SUBTRACT 2	5.00	CARTRIDGE
ADVENTURE (PIRATE)	12.00	CARTRIDGE
ALIEN ADDITION	6.00	CARTRIDGE
ALLIGATOR MIX	6.00	CARTRIDGE
ALPINER	10.00	CARTRIDGE
ASCOT STAKES	2.50	CASSETTE
BATTLESTAR ATTACK	2.50	CASSETTE
BEGINNING GRAMMAR	5.00	CARTRIDGE
BIGFOOT	12.00	CARTRIDGE
BLACKJACK & POKER	6.00	CARTRIDGE
BLASTEROIDS	2.50	CASSETTE
BLASTO	5.00	CARTRIDGE
BOMBS AWAY	2.50	CASSETTE
BOUNCER	3.50	CASSETTE
BURGERTIME	12.00	CARTRIDGE
CHARACTER GENERATOR	3.00	CASSETTE
CHISHOLM TRAIL	5.00	CARTRIDGE
CONNECT FOUR	8.00	CARTRIDGE
DEFENDER	12.00	CARTRIDGE
DEMOLITION DIVISION	6.00	CARTRIDGE
DIGDUG	13.00	CARTRIDGE
DIYAD	2.50	CASSETTE
EARLY LEARNING FUN	5.00	CARTRIDGE
EARLY READING	5.00	CARTRIDGE
ESPIAL	10.00	CARTRIDGE
FACEMAKER	12.00	CARTRIDGE
FUN-PAC	2.50	CASSETTE
HANGMAN	10.00	CARTRIDGE
HOME FINANCE DECISION	10.00	CARTRIDGE
HOP-IT	2.50	CASSETTE
HOPPER	10.00	CARTRIDGE
HOUSEHOLD BUDGET MANAGEMENT	6.00	CARTRIDGE
HUNT THE WUMPUS	5.00	CARTRIDGE
INDOOR SOCCER	8.00	CARTRIDGE
INVADERS	5.00	CARTRIDGE
INVENTORY MANAGEMENT	12.00	DISK
INVOICE MANAGEMENT	12.00	DISK
JAWBREAKER II	8.00	CARTRIDGE
M.A.S.H.	12.00	CARTRIDGE
MAILING LIST	12.00	DISK
METEOR MULTIPLICATION	6.00	CARTRIDGE
MIND CHALLENGERS	6.00	CARTRIDGE
MINI-MEMORY	45.00	CARTRIDGE
MINUS MISSION	6.00	CARTRIDGE
MISSILE ALERT	2.50	CASSETTE
MOON MINE	10.00	CARTRIDGE
MOON PATROL	10.00	CARTRIDGE
MULTIPLICATION 1	5.00	CARTRIDGE
MUNCHMAN	5.00	CARTRIDGE
MUNCHMOBILE	8.00	CARTRIDGE
NUMBER MAGIC	5.00	CARTRIDGE
NUMERATION 2	6.00	CARTRIDGE
OLDIES BUT GOODIES 1	3.00	CASSETTE
OLDIES BUT GOODIES 2	3.00	CASSETTE

TITLE	££.pp	FORMAT
OPERATION MOON	2.50	CASSETTE
OTHELLO	8.00	CARTRIDGE
OTHELLO	2.50	CASSETTE
PARSEC	6.00	CARTRIDGE
PEARL DIVER	2.50	CASSETTE
PERSONAL FINANCIAL AIDS	4.00	CASSETTE
PERSONAL REPORT GENERATOR	6.00	CARTRIDGE
PICNIC PARANOIA	6.00	CARTRIDGE
PROGRAMMING AIDS 1	6.00	CASSETTE
PROGRAMMING AIDS 2	10.00	DISK
PROGRAMMING AIDS 3	10.00	DISK
PROTECTOR II	8.00	CARTRIDGE
RETURN TO PIRATE ISLE	12.00	CARTRIDGE
ROBOPODS	2.50	CASSETTE
SAMURAI	4.00	CASSETTE (INCLUDES COMPUTER DUST COVER)
SLYMOIDS	12.00	CARTRIDGE
SNEGGIT	8.00	CARTRIDGE
SPEECH EDITOR	12.00	CARTRIDGE
SPELL WRITER	12.00	DISK
STATISTICS	12.00	CARTRIDGE
SUPERFLY	12.00	CARTRIDGE
TEACH YOURSELF BASIC	3.00	CASSETTE
TEACH YOURSELF EX-BASIC	3.00	CASSETTE
THE ATTACK	3.00	CARTRIDGE
TOAD GRAPHICS	3.00	CASSETTE
TOMBSTONE CITY	5.00	CARTRIDGE
TROLL KING	2.50	CASSETTE
VIDEO CHESS	12.00	CARTRIDGE
YAHTZEE	6.00	CARTRIDGE
ZERO ZAP	6.00	CARTRIDGE

COMPUTE! BOOKS

TI SOUND AND GRAPHICS	9.95	BOOK
PROGRAMMERS REFERENCE GUIDE	9.95	BOOK
GUIDE TO EXTENDED BASIC HOME APPLICATIONS	9.95	BOOK
BEGINNERS GUIDE TO ASSEMBLY LANGUAGE	9.95	BOOK
33 PROGRAMS FOR THE TI	7.95	BOOK
CREATING ARCADE GAMES	7.95	BOOK

SPECIAL OFFER

STARTER PACK 1	3.00	BOOK/CASSETTE
STARTER PACK 2	3.00	BOOK/CASSETTE
GAMES WRITER PACK 1	3.00	BOOK/CASSETTE
GAMES WRITER PACK 2	3.00	BOOK/CASSETTE

TI-99/4A KEYBOARD SCHEMATIC

From the JUNE 1985 newsletter of the TI NOVA SCOTIA USER GROUP

TECHNIQUE

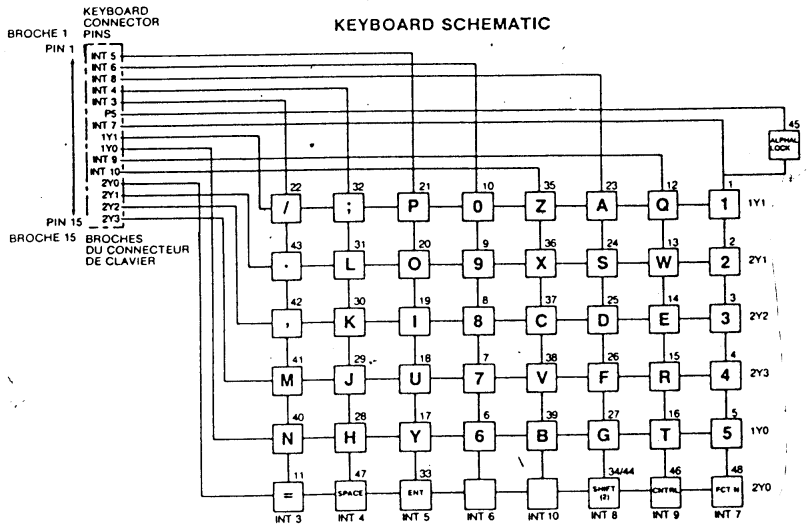
TI99/4A Keyboard

For those of you who have schematics for every section of the TI99/4A computer **except** the keyboard itself, here it is.

Now you can determine which key relates to which pin at the connector. This may be very helpful in making a "lap-keyboard" for use with those games that call for alphanumeric inputs. Make an extra long ribbon-cable lead, add a 15 pin connector and sit back comfortably to play.

The enterprising might want to add a fixed connector receptacle on the joyport side of the console, just behind the joyport itself. If you do, don't forget to draw it up for the rest of us.

[Warning: These modifications negate warrantee rights]



CONFIGURING TI FORTH TO YOUR SYSTEM (Part 1)

By MIKE RICCIO

Taken from the January 1985 newsletter of the PHILADELPHIA AREA USER GROUP

TI FORTH, a powerful language for your TI computer, was released to the Users' Groups by Texas Instruments as PUBLIC DOMAIN. Since then, almost all the people in our group who can run FORTH have obtained a copy. TI FORTH requires the following equipment:

CONSOLE AND MONITOR (OR TV)
DISK CONTROLLER
AT LEAST ONE DISK DRIVE
32K MEMORY EXPANSION
EDITOR/ASSEMBLER MODULE

{There is an Extended BASIC loader for FORTH in the Public Domain, and I understand that a MiniMemory version may also be available. I have copies of FORTH, with manual and disk, for £10. Contact me if you would like a copy, or for further information. PB}

The TI FORTH disk is set up for certain parameters, which assume that you have the following:

One single sided, single density disk drive
An RS232 compatible serial printer

This article will show you how to modify these requirements to suit your own configuration, and also set up FORTH so that it loads your most-used options very rapidly.

It's probably too late, but BEFORE YOU DO ANYTHING WITH YOUR TI FORTH DISK, MAKE A BACK-UP COPY OF IT FIRST! Your FORTH disk has some valuable information on it, and many FORTH commands could damage this, so PLEASE make a back-up copy! Use your Disk Manager module to make a back-up copy of FORTH, but ONLY copy it on to a SINGLE SIDED/SINGLE DENSITY format disk otherwise it will not be compatible with FORTH.

{You can use a double-sided drive provided that you tell Disk Manager that there is only one side to the disk! PB}

The first step is to enter FORTH. Do this by powering up your system. Insert the Editor/Assembler module. Turn on the console. Press any key, then press 2 for Editor-Assembler. Now press 3 for Load and Run. Place the FORTH disk in drive 1 and type in: DSK1.FORTH and press ENTER. You have just loaded FORTH!

After you load FORTH, the first thing you see is 3 columns of words with dashes in front of them. These are known as the loading options. Below them you see the words TI FORTH and a flashing cursor. Whenever you see the flashing cursor, it means FORTH is waiting for you to tell it what to do. As with BASIC, you press the ENTER key when you are done typing.

The remainder of this article will deal with configuring FORTH, so if this is the first time you have used FORTH, please, open the manual and read the first three chapters. If you've done this, then read on...

FAST LOADING FORTH

The loading options give you more commands of many varieties, but as you know it takes a LONG time to load. We're going to change all of that. Right now you must decide which options you are going to use the most; load these by typing their names as they appear on the list (don't forget the dash!) and press ENTER. You can load more than one at a time by separating them with a space. You must also decide which editor you like better, the 40 column or the 64 column, because you can only load one. After loading the options, type in the following line:

```
-BSAVE ' TASK 51 BSAVE
```

(' is the apostrophe - FCTN O)

That was the hard part! Now all you have to do is to edit SCREEN 3, the "Boot" screen. Type in the following lines, pressing ENTER after each:

```
3 CLEAR
FLUSH
3 EDIT
```

You are now in FORTH's editor. Now enter what you see below:

```
0    ( WELCOME SCREEN ) BASE->R  HEX  10 SYSTEM
1    0 0 GOTOXY ." LOADING TI FORTH... " CR 10 83C2 C!
2    DECIMAL 51 BLOAD  16 SYSTEM  MENU
3
4    1 VDPMDE !
5    0 DISK_LO !
6
7    : SIZE   SP@ HERE - . ." BYTES FREE" ;
8    : PAGE   0 0 GOTOXY CLS ;
9    : BYE    MON ;
10
11   R->BASE
12
13
14
15
```

Now press BACK (FCTN 9) and enter the following lines:

```
FLUSH
TEXT
COLD
```

FORTH should re-boot as if you had just loaded it. If not, re-enter FORTH using the back-up disk, load the editor, and check your typing on SCREEN 3. By the way, we've also added 3 new commands to FORTH:

```
SIZE   tells you how much memory is left
PAGE   acts like CALL CLEAR in BASIC
BYE    exits TI FORTH
```

Next time, we'll learn how to modify FORTH for more than one disk drive, for a parallel printer, and correct an error made by TI.

~~~~~~

CONFIGURING TI FORTH TO YOUR SYSTEM (Part 2)

By MIKE RICCIO

Taken from the February 1985 newsletter of the PHILADELPHIA AREA USER GROUP

DRIVE SETUP

Last month I left the fourth line on screen 3 blank for a reason, and this is it. Type the following lines pressing enter after each:

```
FLUSH
3 EDIT
```

Refer to the following table for what to put on the fourth line.

TYPE THIS:		IF YOU HAVE THIS:
90 DISK_HI !	90 DISK_SIZE !	One SS/SD disk drive
180 DISK_HI !	90 DISK_SIZE !	Two SS/SD disk drives
360 DISK_HI !	90 DISK_SIZE !	Three SS/SD disk drives
180 DISK_HI !	180 DISK_SIZE !	One DS/SD disk drive
360 DISK_HI !	180 DISK_SIZE !	Two DS/SD disk drives
540 DISK_HI !	180 DISK_SIZE !	Three DS/SD disk drives

Please note: FORTH is not set up for double density drives.

Now press BACK (FCTN 9) and type FLUSH and press ENTER.

FORTH ERRORS

The TI FORTH disk contains 2 resident errors. The first is on screen 72. Type:
~~~~~~

72 EDIT

(From now on, when I say to edit a screen, type its number, then EDIT and press ENTER. Also, line numbering on screen starts at 0 and goes to 15).

Look at line 5. It may currently read:

```
PAB_ADDR @ VSBW 1 PAB-ADDR @ 5 + etc.....
```

Change it to:

```
PAB-ADDR @ VSBW 1 PAB-ADDR @ 5 + etc.....
```

The second is on screen 58. If you did not do the fast loading FORTH last month, then you can correct the error by editing screen 58, and changing line 10 from:

```
VDPMDE @ 4 ( IF SMTN 80 0 VFILL 300 ! SATR ! ENDIF
```

to

```
VDPMDE @ 4 ( IF SMTN 80 0 VFILL 300 ' SATR ! ENDIF
```

Otherwise, if you did modify your disk, edit screen 89 to look like this:

```
( CORRECT WORKING SSDT ROUTINE 19JAN85 MR )

HEX

: SSDT2 DUP ' SPDTAB ! 800 / 6 VWTR SATR
        20 0 DO DUP )R D000 SP@ R) 2 VMBW
        DROP 4 + LOOP DROP VDPMDE @ 4 ( IF
        SMTN 80 0 VFILL 300 ' SATR ! ENDIF ;

DECIMAL

' BRANCH CFA ' SSDT ' SSDT2
OVER - 2- OVER 2+ ! !
```

And edit screen 3 so that line 6 reads:

```
89 LOAD
```

Finally, press BACK, type FLUSH and press ENTER.

PARALLEL PRINTER

FORTH is currently set up for a printer configuration of: RS232.BA=9600. But if you have a parallel printer (PIO), do this to make it compatible with FORTH.

Edit screen 72. Change line 4 from:

```
SET-PAB OUTPT F-D" RS232.BA=9600" OPN 3
```

To:

```
SET-PAB OUTPT F-D" PIO" OPN 3
```

Press BACK, type FLUSH and press ENTER.

JUST FOR FUN

Type this line and press ENTER:

```
1 SCRN_WIDTH !
```

Now try typing things. Is anything strange happening ?

By the way, if you're using the 64 column editor, after editing a screen type TEXT and press ENTER to return to normal 40 columns.

Well, that's all for this time. Some of the information presented here may be too technical for you, but don't worry, next time we'll start a tutorial on how to become well versed in FORTH. Till then, live long and program!

A LOOK AT PROGRAMS

By R. A. GREEN

Taken from the November 1985 newsletter of the OTTAWA TI-99/4 USERS GROUP

There are seven different ways to store programs in the TI-99/4A. In this article we will have a look at each of these seven forms and at how they are used.

Everyone is familiar with the form used by TI BASIC to store programs on cassette or disk. It's identified as "PROGRAM" in the disk catalogue. It is created or stored by the BASIC SAVE command and loaded by the BASIC OLD command. This is the only way that TI BASIC uses to store your programs.

Extended BASIC can, and usually does, use the same form as TI BASIC to store programs. In fact, Extended BASIC can use TI BASIC programs. There are, however, two other forms that XB uses. Both these forms can only be used to store programs on disk.

If you have the 32K Memory Expansion, you can write an XB program which is too large to store in the usual format. XB will store these large programs in an "INTERNAL VARIABLE 254" file. The usual "SAVE" and "OLD" commands are used to store and load these programs.

The third form used by XB is the "merge format" stored in a "DISPLAY VARIABLE 163" file. This form is created when the "MERGE" option is specified in the "SAVE" command, and is loaded by the XB "MERGE" command. The beauty of merge format is that when it is loaded it does not necessarily overwrite the program in memory. The MERGE command does just that - it merges the new program (or program segment) with the program in memory according to the line numbers.

Now, we get to the good stuff, Assembler language programs. There are three forms for an assembler program: tagged object, compressed tagged object and memory image.

Tagged object is stored in a DISPLAY FIXED 80 file on disk only. All program data is in hexadecimal so that it can be edited by the E/A editor. Tagged object can be loaded via CALL LOAD in XB, option 3 on the E/A menu, option 1 on the MM menu or by CALL LOAD in TI BASIC when either the E/A or MM module is used. The program can be "absolute" or "relocatable". An absolute program must always be loaded at the same place in memory. A tagged object program may have references to other programs or subroutines. The loader will resolve these external references, except for the XB loader.

Compressed tagged object is very nearly the same as tagged object except that the program data is stored as bytes rather than as hexadecimal digits. Compressed tagged object loads faster than regular tagged object as you would expect. The XB loader cannot load compressed object.

Tagged object, in either form, is produced by the Assembler when it assembles a source program.

The "memory image" form of assembler programs is the most compact and the fastest loading. It can be stored on cassette or disk. It is identified as PROGRAM in the disk catalogue (just like a BASIC program). Memory image programs can be loaded by option 5 on the E/A menu or option 3 on the TI Writer menu (and I assume, by Multiplan, although I have never tried since I don't have Multiplan). It should be noted that there is one slight but important difference between how the E/A calls a memory image program and how TI Writer does. TI Writer blanks the screen just before calling the program and the E/A does not. This means the program must turn the screen back on or nothing will show. Memory image programs are created by a Utility program (one is provided on the E/A disk).

A PROGRAM file, containing an Assembler memory image or a BASIC program, can be read or written to any input-output device with a single I/O operation. This is one of the reasons they load so quickly.

There is a restriction on the size of an Assembler memory image program of >2400 bytes (9216 decimal). However, the E/A and TI Writer modules will load multiple memory image files to make a program of any size. They use the convention that the file name of the second and following files is obtained by incrementing the last digit or letter of the previous file name.

For example, the TI Writer editor consists of two memory image files: EDITA1 and EDITA2.

As a matter of interest, the ADVENTURE, Tunnels of Doom, Personal Record Keeping, Statistics and Personal Report Generator modules use a memory image or PROGRAM file for their data bases. The fact that memory images can be saved or loaded with a single I/O operation makes them attractive for such uses.

A lot of the Assembler language games that are circulating around are in the memory image format so let's look closer at them. Assembler memory image files have a three word header followed by the data to be placed in memory. The three header words are:

1. This word is a "flag". If it is not zero (i.e. >FFFF) then this file is not the last in the multi-file program. For example, the flag word for EDITA1 is >FFFF indicating that there is another file called EDITA2; the flag word in the EDITA2 file is >0000 indicating it is the last file and there is no EDITA3.

2. This word is the length of the memory image in bytes, including the six byte header.

3. This word is the CPU memory address where the memory image is to be loaded.

Execution of a memory image program always begins at the first byte of the first segment loaded.

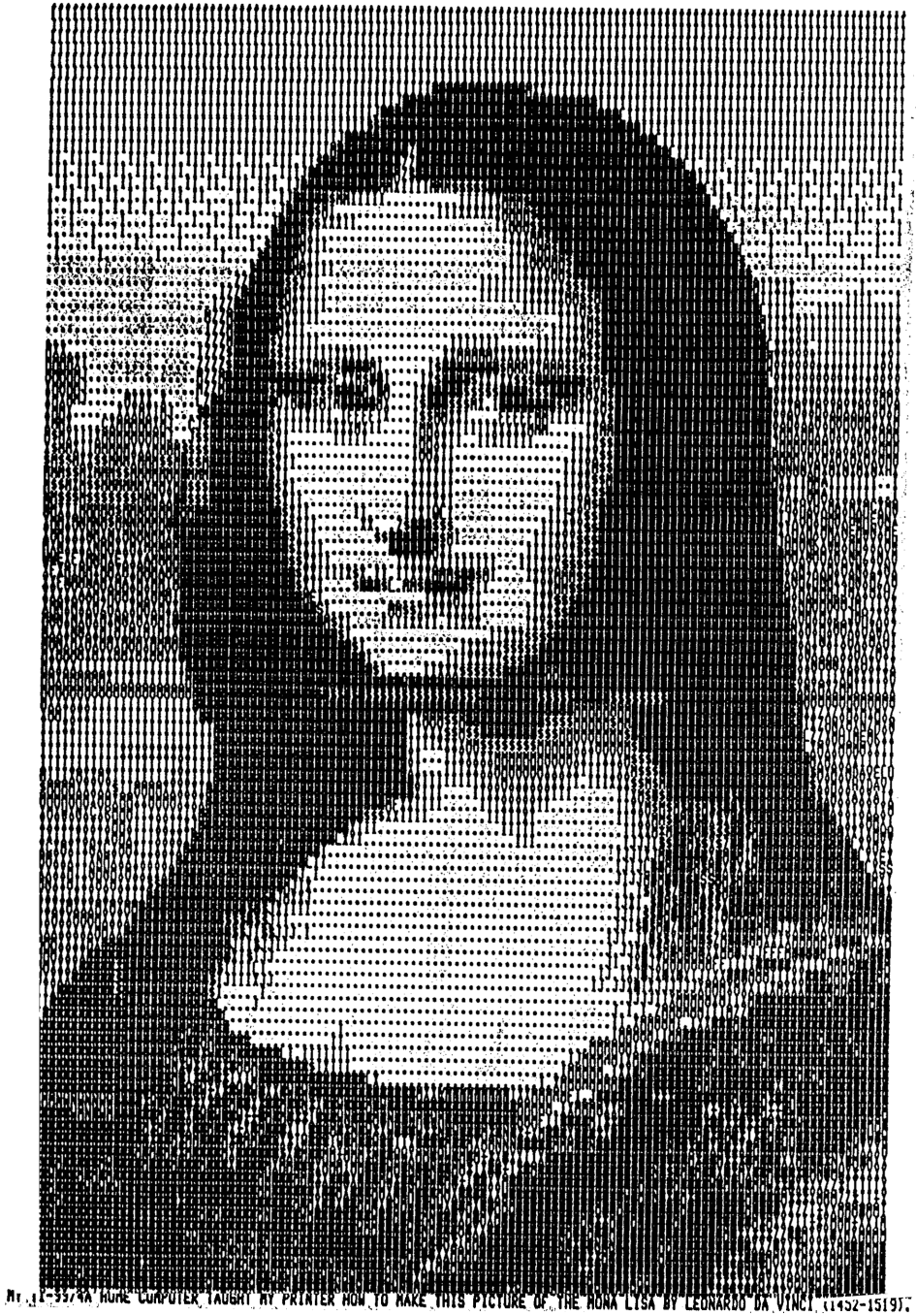
Finally, the seventh form for programs. This form is created and loaded by EASY BUG of the Mini Memory Module. It can be written only to cassette and is a memory image, but is slightly different from the E/A memory image file. The EASY BUG memory image program can consist of only one segment. The header on the EASY BUG format is two words, as follows:

1. This word is the CPU memory address at which the memory image is to be loaded.

2. This word is the length of the memory data, not including the four header bytes.

If this whole thing is too complicated - maybe a table showing all the options will help.

FILE TYPE	CONTENTS	MODULE	DSK	CS
PROGRAM	BASIC Program	Console	YES	YES
PROGRAM	BASIC Program	XB	YES	YES
INT/VAR 254	BASIC Program	XB	YES	NO
DIS/VAR 163	MERGE Program	XB	YES	NO
DIS/FIX 80	Tagged Object	XB	YES	NO
DIS/FIX 80	Tagged Object	E/A	YES	NO
DIS/FIX 80	Tagged Object	MM	YES	NO
DIS/FIX 80	Compressed Object	E/A	YES	NO
DIS/FIX 80	Compressed Object	MM	YES	NO
PROGRAM	E/A Memory Image	E/A	YES	YES
PROGRAM	E/A Memory Image	TIW	YES	YES
PROGRAM	MM Memory Image	MM	NO	YES



Printed by a TI99/4a

CLOSE FILE

MORE HELLOS

Since writing the Editorial, NICHOLAS FROST and BRIAN BENNETT have joined our ranks, and the technically-minded among you can look forward to at least one article from Brian, which will probably appear in the April issue. I say "probably", because April may have more than its fair share of eye-opening material, which will leave little room for further revelations.

BRIAN ABRAMS rang me to tell me that he had successfully completed the 32K RAM project presented in the last issue, (at the time of writing I have not yet received the updated diagram from GRAHAM WOLSTENHOLME, although I did delay publication as long as I dared).

DAVE HEWITT (see also the BULLETIN) is offering another service to OTIUsers: he will assist in the production of the 32K RAM unit as described last issue either by building and installing the 32K to a negotiated price (less than £50 I would guess)provided that he is absolved from any financial obligation should any fault occur, or by using materials which you provide (assuming that you can obtain the necessary chips etc. more cheaply than he can!).

Dave also notified me of a small bug in one of the programs which were sent in by JAMES STRINGFELLOW (published in V2.8): on page 17, statement 148 has a 136 in its DATA statement (about the fourth number along). This ought to be 144 - it enables the full range of ASCII characters to be defined, up to and including ASCII 143 that is.

That reminds me: I can make available on disk the programs which James sent in, and where practical also on tape. Feast your peepers on James's latest offering in this issue - I am beginning to lust after hi-res graphics on my printer!!

DAVE CARR's peripheral expansion problem resolved itself spontaneously, but obviously if anyone can suggest any explanations I am still happy to receive and possibly publish them.

WAITING FOR BAUDOT (with apologies to Becket...)

On a page somewhere in this issue you will find a clever portrayal of a reasonably well-known lady, distributed by ED YORK of the CIN-DAY USER GROUP in his newsletter. This month's flashy competition is to find a suitable (and printable) caption for what you think might be going through the young lady's mind. The challenge is thrown open to any OTIUser or member of the family of an OTIUser. My problem is trying to think of a suitable prize!

It will probably be software/literature, with a choice for the luck(less) winner.

Finally, does anyone have any useful information on how to make a Tandy DMP 105 printer compatible with the hi-res graphics output of GRAPHX ? Let me know and I will publish/pass on the information.

It is now 4.22 am and sleep beckons. Yawn...

32K RAM Project

There was unfortunately an error in part of the veroboard circuit for the 32K RAM. Shown below is the correct wiring sequence for the GND wire. Please note the additional break in the copper strip between GND in and PIN 15.

Hopefully anybody who attempted the project will have noticed the error, when comparing the circuit with the Australian Article. I hope this did not cause problems for anybody.

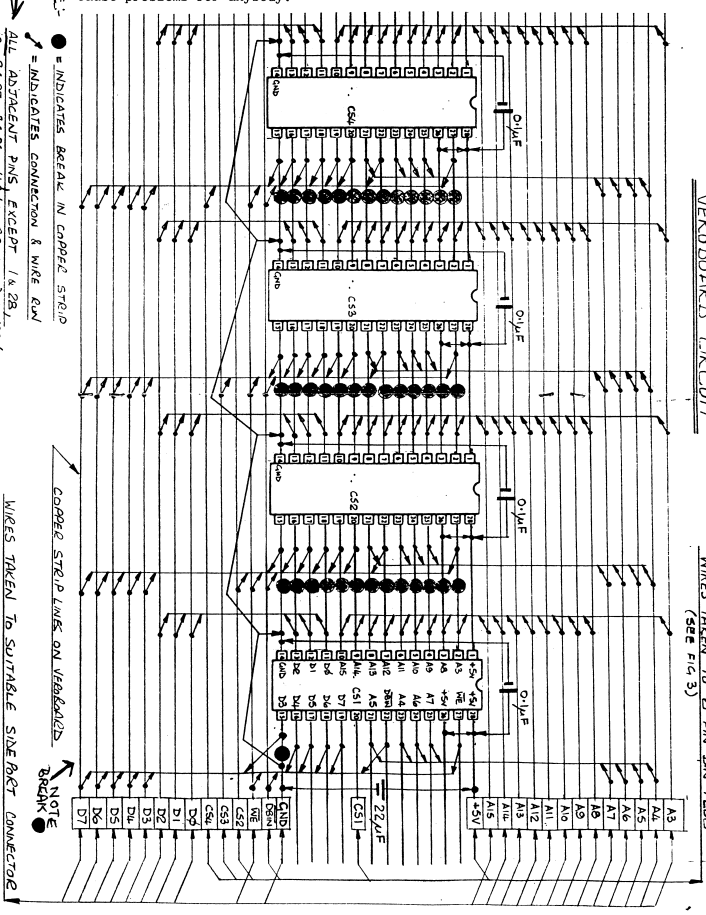
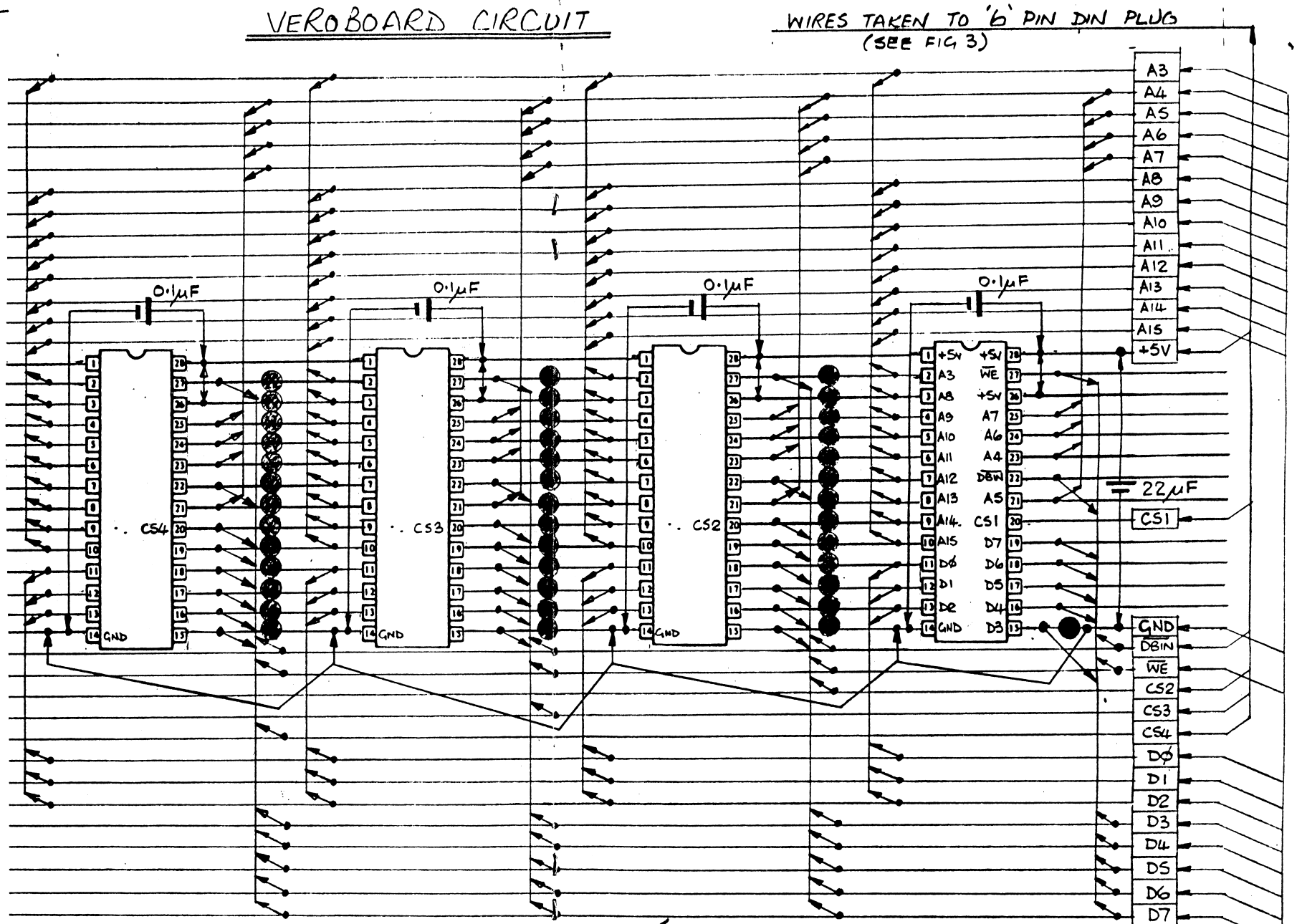


FIG 1

There was unfortunately an error in part of the veroboard circuit for the 32K RAM. Shown below is the correct wiring sequence for the GND wire. Please note the additional break in the copper strip between GND in and PIN 15.

Hopefully anybody who attempted the project will have noticed the error, when comparing the circuit with the Australian Article. I hope this did not cause problems for anybody.



NOTE:- ● = INDICATES BREAK IN COPPER STRIP

↗ = INDICATES CONNECTION & WIRE RUN

ALL ADJACENT PINS, EXCEPT 1 & 28, ie 2 & 27, 3 & 26, HAVE BREAK BETWEEN THEM IN COPPER STRIP. THIS IS NOT SHOWN ON ABOVE LAYOUT.

FIG 1