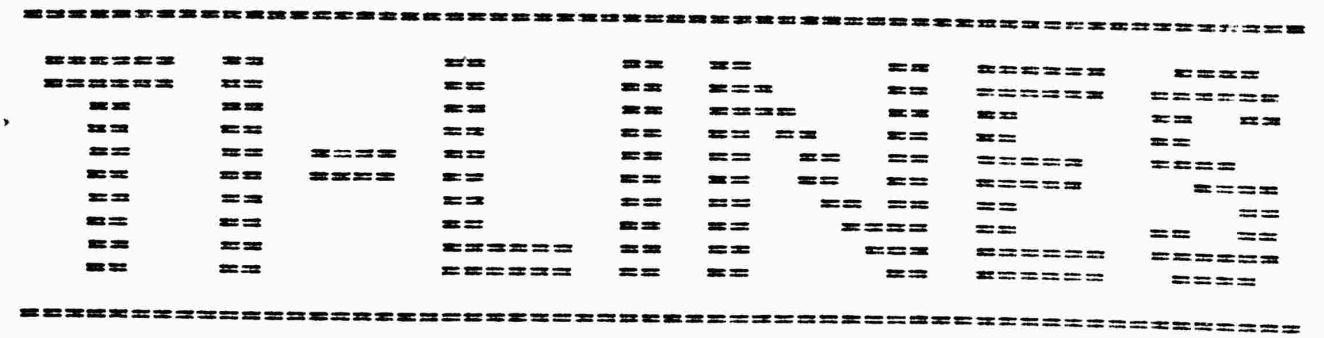


Volume 1, Issue 13 **April 1st., 1985**

TI-LINES is produced and published by:

Peter G. Q. Brooks
61 The Avenue
Kennington
OXFORD
OX1 5PP

Tel: OXFORD 730044. Please do not phone after 10 pm

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

TI-LINES is available on Associate subscription to Users resident outside Oxfordshire, for £10 p.a. Overseas subscriptions are by arrangement. 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, or in a form which is as legible as possible. Art work 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.

I n d e x t o a r t i c l e s

S E E Y O U A T T H E B R I G H T O N S H O W

EDITORIAL

REFORMAT

I have finally got my Smith Corona TP-I up and running, and by the time that you get to read this I should have been able to obtain the necessary additional bits in order to reproduce accurately the full 96 characters of the ASCII set. This should lead to an increase in legibility on the part of TI-LINES (discounting the vagaries of the photocopier) and will also form the basis of a trial run of much of the formats/typefaces to be used in the Grand Booklets Scheme.

Talking of photocopier vagaries, my apologies for the foul copy produced last issue. CHARLES LACEY, who has kindly allowed me to use his machine to produce TI-LINES, had received a visit from the engineer a day or so before I arrived, and the conscientious service which the technician had performed had left a long scratch or something on the drum, with the inevitable result that a long black line appeared on every copy which the machine made. As TI-LINES was running close to appearing a month late again, I decided to "publish and be damned", which is why you received a grotty copy around the 18th of March.

This month I hope will make up for everything. To counteract the double dose of Yurgh! which being April and the 13th issue will doubtless invoke, I have thrown a large number of extremely good goodies together. This will also, I hope, have the knock-on effect of persuading some of you to put pen to paper. If you think about it, just one article of two A4 pages or so from every OTIUer once every 12 months will be enough to fill more than half of every issue. ALLEN BURT has fulfilled his quota for the next two years, as has TONY RALPH, and VIV COMLEY has provided a much-sought-after article on connecting the TI to an RGB monitor which must entitle him to take at least a two-year breather. SYD MICHEL started a ball rolling last issue which I hope will be picked up and run with, if not by him then by some other stalwart. DAVID BROWN has earned his wings many times over, and I would like to feature more material from PAUL DUNDERDALE if I could (hint, hint, Paul!).

I am also going to break with my tradition of not quoting wholesale from other newsletters (on the grounds that the same information then gets passed round and round without generating anything new) because of a number of items which have come to my attention courtesy of User groups in the States and in Canada. I have apparently adopted the same attitude that they have with regard to the information they present: the originator and the "audit trail" of secondary sources are given wherever they are available. This could lead to each TI-LINES article being preceded by a preparatory article detailing where all the data has its origin. I have visions of two-thirds of each issue being given over to source details.....

The new format could also lead to a thinner issue each month from about JUNE, as it compresses the text considerably. Older TI-LINES crammed only about 3960 characters per page into an issue, while the new format will attempt to put no less than 5600 per page. That works out to about a 40% reduction in capacity while still retaining the same content. I hope that you will like it. What is more, I hope that I can do it!

OOPS! SORRY SYD!

I carefully typed up SYD MICHEL's article last issue and dumped it out to the printer without looking too closely - until after I had collated all the copied sheets. Then I saw the dreaded £££s instead of the appropriate hashes. My only query about the program on page 14 last issue was the use of CHAR as a variable name - I thought that it might have been seen by the system as a Reserved Word, and therefore Taboo, but it seems not. You can use subprogram names as variable names with impunity, yet off the top of my head I could have sworn that it should have produced an error. You learn (or relearn) something every day...

~~~~~~

APRIL FOOL

It is the custom to include at least one spoof item in any publication bearing an April copy date, and I missed the boat last year. This year, however, I will join the Fools and tell you that there is one spoof item in this issue. No prizes for guessing which one it is!

~~~~~~

PIO ON TI-OH

I have always been given to understand that TI's idea of a parallel interface is not Centronics idea of one. In fact, you will find a quickie diagram of a harware "fix" for the PIO lead elsewhere in this issue, courtesy of DAVE KNOTT of CENTRONICS, provided through the auspices of MALCOLM HEDLEY. Having already burned my fingers when I tried to improve on someone else's hardware information I have left well alone, and the diagrams are as received from Malcolm and Dave.

Malcolm has also advised me that EPSON printers will not require this fix, as TI engineered the PIO to suit the Epsons apparently. I had wondered about this compatibility, as I had a report from newly-joined OTIUer J. MAYFIELD that his first Epson would not work with the TI, and not until he had returned both printer and cable to the supplier for them to sort out did he get a replacement printer and success. I have very limited hardware knowledge in this area, but if anyone has had problems with wiring their printer into the RS232 card they could do worse than get in touch with me. (They COULD do worse, but not much!)

~~~~~~

THE END IS NIGH!

After April comes May, and after May comes Volume 2 of TI-LINES. Yep, the first fourteen issues will have been produced by the end of May, and then it's pocket-dipping time. One or two of you have already dipped, for which many thanks, and if hordes of you want further drivel plummeting through the letterbox within say two months of the cover date, then don't forget to renew your subscription.

Because of financial considerations I will have to appear mean and vicious and request your renewal BEFORE THE END OF MAY, so that I know how many copies of the first issue of the second volume to produce. Obviously if you renew after JUNE you will still receive all the issues in the then current volume, but it
~~~~~~

will help me enormously if I have only a few late renewals to cope with. I will be putting my neck on the chopping block and making 70 copies of V2.1 anyway, just in case, and probably of V2.2, V2.3, etc., until I realise that only three of you have decided to subject yourselves to further mental and ocular anguish.

~~~~~~

RIFE RUMOURS

The word on the grapevine was that MYARC were about to take over the ailing CORCOMP, who filed last year for a special form of bankruptcy. Two sources confirmed each other, and even 99er/HCM apparently made veiled comment on the event. Quite how this would have affected the sales of equipment in this country is hard to predict.

~~~~~~

HELLO FOLKS

We have some more OTIUers to greet. A warm welcome to IAN JAMES, HENRY CLARK, Mr J. MAYFIELD, PHILLIP MARSDEN, GORDON PITT, FRED PABIAN of the MILWAUKEE AREA USER GROUP in the States, FRANCIS X. GASTON of the SASKATOON TI USERS in Canada, ERNST NOVAK of the PHILADELPHIA AREA TI USERS, and BOB MACK (User group unknown at the time of writing).

~~~~~~

OH NO, NOT AGAIN

Time has whistled by and I have still not managed to catch up on the backlog of material waiting to be published. With the aid of Mrs JENNIFER KEANE I have been able to get a lot of the material typed up (on a typewriter which lacks the basic items like an asterisk!) but Easter is looming and I am away for about ten days trying to sort out a future for the TIHOME Software Collection. I have decided therefore to (yet again) hold off publishing this issue of TI-LINES until I can sit down and put it together properly, rather than rush into it. If Mrs Keane can manage a few more sessions we should have part of the problem licked anyway, so catching up in future should be less of a hardship. Distant rumblings suggest that I may be forced to change my address again shortly, but I will inform readers obviously as soon as I know anything definite.

~~~~~~

R. I. P.

A couple of issues back I praised a fortnightly magazine called MICROMART, which advertised secondhand computers and related items. It has now ceased to be published. Apparently it didn't manage to pay its way.

THE FORMAT OF DOOM

STEPHEN SHAW understands that someone across the Big Pond may have details about the format of the database for TUNNELS OF DOOM. Can anyone help him out ? You can contact him direct at 10 ALSTONE ROAD, STOCKPORT, CHESHIRE SK4 5AH or on 061 432 6097 (mind you don't wake George).

EXTENDED BASIC

A rumour has reached me that a company in the States has obtained a license to produce TI's Extended BASIC. Quite when (or if) this product will hit the streets is not certain.

WOULD THE REAL BULLETIN BOARD PLEASE STAND UP ?

Elsewhere in this issue you will find an extract from a letter from ALAN DAVEY, a recently-joined OTIUer who is looking to perhaps operate a "real" bulletin board - the type that can be accessed via the telephone. I know that some of our members are "wired up", and in the States most of the groups seem to have some form of bulletin board involving the TI-99s. If the phone charges were not so high I think that more people would invest in the necessary equipment, and usher in the "wired society" which was talked about in the magazines a few years ago.

MEMBERSHIP CARDS

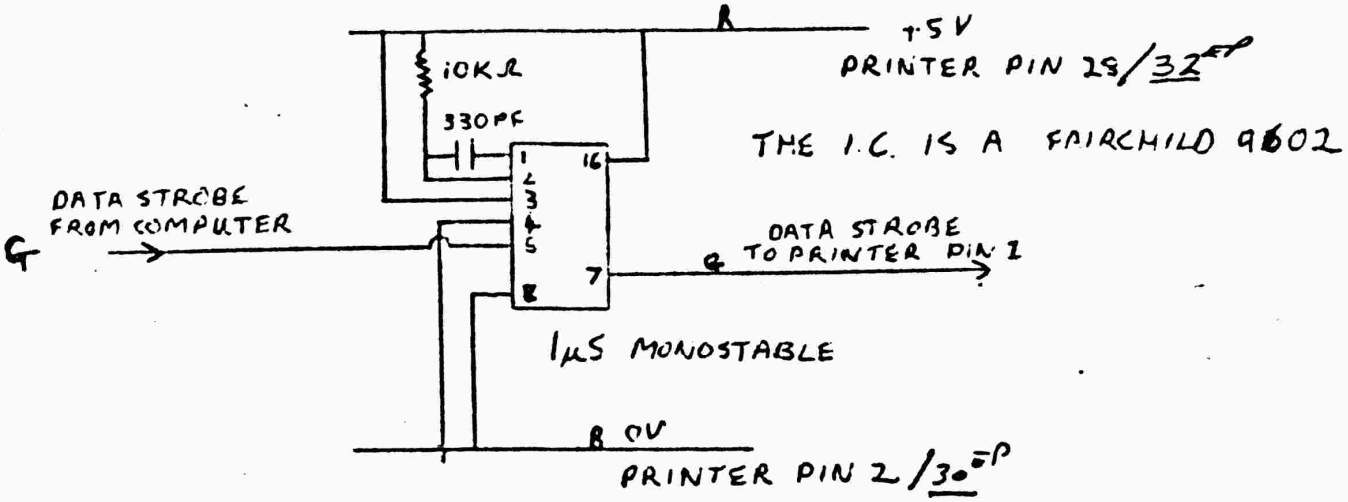
I have had a set of OTIU membership cards made up ready for 1985/86. They will be distributed at the beginning of the new subscription year to those members who wish to renew their subscriptions and should serve better as a form of identification at meetings than those daft bits of card that I have recently sent out!

C O N T A C T S

JOHN MAYFIELD	13 Orchard Close, Spencers Wood, Reading RG7 1EJ Tel: 0734 883860
JAMES STRINGFELLOW	23 rue Pasteur, 78700 CONFLANS Ste H., France
JOHN BINGHAM UK Contact Address:	Rygghagen 7B, 4070 Randaberg, Norway. 11 New Road, Esher, Surrey, KT10 9PG

CENTRONICS

I have enclosed a circuit which should enable your printer to work on your Texas computer. Below is the circuit diagram for it.

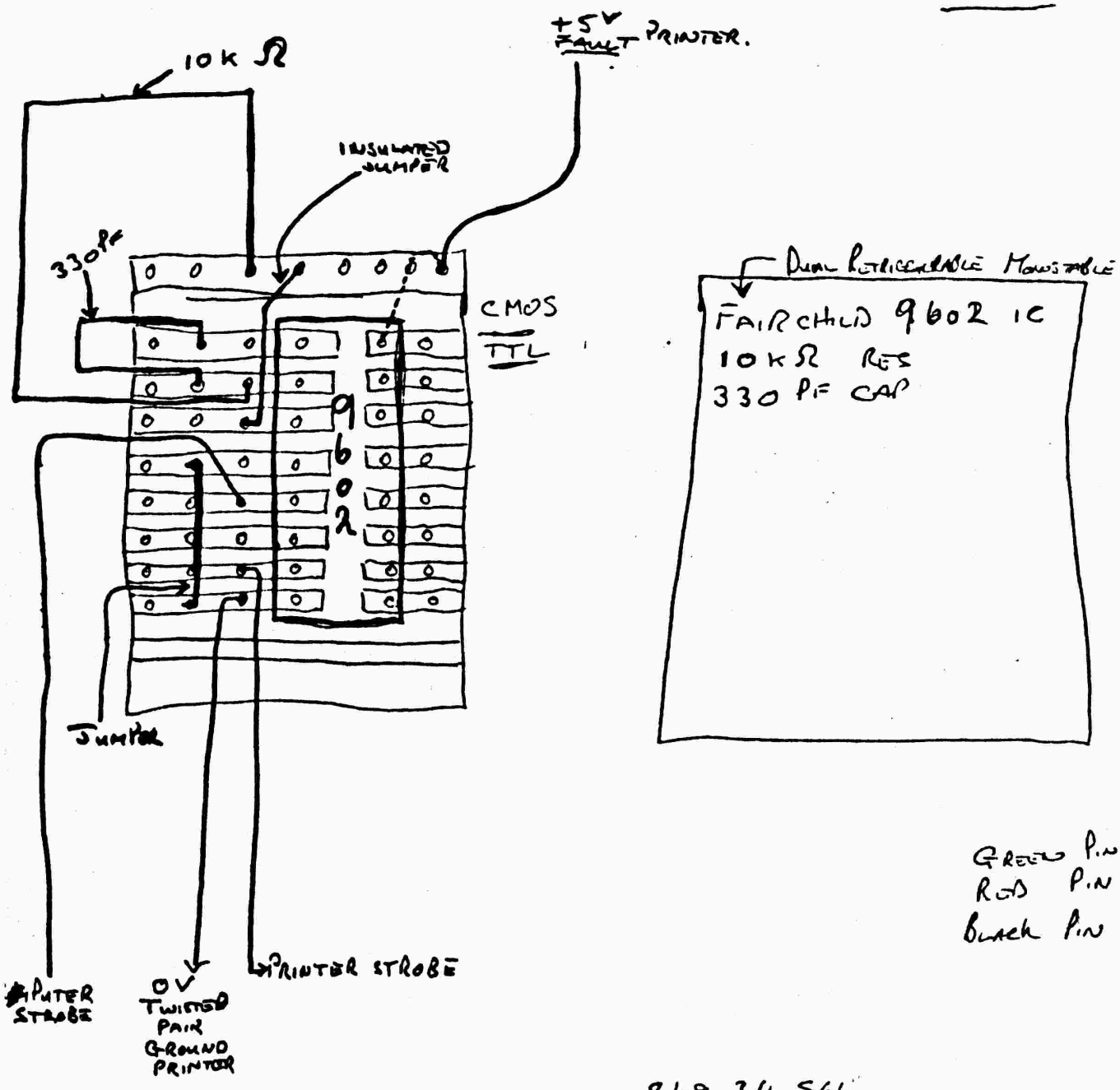


Regards
Dave Knott

TI-99/4A PIO HARDWARE FIX

From DAVE KNOTT of CENTRONICS, and MALCOLM HEDLEY

74_S123



869 34 564

LETTERS

Dear Pete,

I have been investigating various aspects of the PRK module and its resident subprograms using your articles in TI-LINES and a copy of TI's own booklet on the subject. One of the tasks I hope to achieve soon is to write an ENHANCED BASIC program to take over the printing of reports, since the computer can lock up under certain conditions when printing from PRK. I am not yet in a position to give details of my findings (or write an article, should you be interested) but I think I ought to make several points, in case you were not yet aware of them (not likely!).

Firstly, I don't know if you have the addendum to the PRK manual, which points out several problems that might arise. (If you have the addendum, then please accept my apologies for repeating something which is obviously old hat to you). The problem can occur in files which have a field of CHAR-type with a length of one - such as fields 11 and 12 of the CLUB-85A example in the February 1985 edition of TI-LINES.

This sequence of events leads up to the error:

1. Enter a complete record ("page").
2. Select CHANGE PAGE mode, and enter the number of the page you entered in step 1.
3. Keep pressing ENTER until you reach the field of MEM-TYPE.
4. Press FCTN BACK, as if you had changed your mind about editing that page.
5. Then try and display or change that page. The computer will lock up with the message * INCORRECT STATEMENT IN 10350.

From experimenting with this crash (a very boring process involving repeated turning off of the console!) I believe I can go some way to explaining its cause. Apparently, pressing FCTN BACK in the first character of such a field causes that character to be lost. The field now contains a null string.

When the computer next accesses that particular field, it tries to display a null string with CALL D(). This, unlike PRINT "", is not allowed, and the computer crashes.

(I would suspect, therefore, that the PRK module consists solely of the ENHANCED BASIC subprograms and an ENHANCED BASIC program , which utilizes these, put on a chip. This would also explain the extraordinary slowness of the PRK when it comes to tasks like alphabetizing).

I hope that this warning may spare the readers of TI-LINES from the frustration of data loss while experimenting with your example program .

Now for the (hopefully) more interesting problem of printing PRK files. Shortly, I hope to have my own ENHANCED BASIC program for this purpose, which should be able to cope with lines of 136 characters for decent report-format printouts, rather than the 80-column limit imposed by PRK. However, I would be interested to hear from you whether you have discovered anything about the conditions under which the various errors occur - I have had crashes with * MEMORY FULL in 4400, 5880 and 5220 whilst sometimes the PRK is quite happy to print out a relatively large file completely in report format. Nor have I had any problems producing screen-dumps when asked to "PRESS P TO PRINT".

I hope I have not wasted your time with this letter. I would much appreciate any (hopefully) helpful or encouraging comments you'd like to make.

Yours,

Jeremy Bygott

(Jeremy Bygott - OTIU Member No. 15)

ALAN DAVEY writes about his intention to set up a Bulletin Board:

I have an auto-answer modem - the INTERLEKT PORTMAN model - which is approved by BT (British Telecom). Auto-answer is necessary if you want the computer to answer the phone all by itself. BT approval (a green ticket) is necessary if you do not want to be held responsible for any faults which BT's engineers may find in your area. They can become very unpleasant if they catch you with any non-approved (or red ticket) items connected to the phone system!

The Portman has three baud rates: 300, 1200, and 1275 (Prestel, which is also on 300).

A three baud rate modem is not essential (and can be very expensive); usually a single rate (300 baud) modem will be quite adequate, and you can still have auto answer.

300 baud is the most popular rate for accessing Bulletin Boards in this country, and is also the easiest for TI-99/4A Users (the more so with the TERMINAL EMULATOR II - TEII).

As for 1200 and 1275, they need special software which is not available here. In the States there is software for both "access to" and "Bulletin Board" form.

This brings me to my own particular interest: I would like to run a Bulletin Board for 99/4A Users on 300 baud to start with, and perhaps progressing one day to running all three rates, possibly on a weekly rotation.

I am also hoping to get into Robotics (with a bit of luck, and a bit of help, too), and into sensors and relays allowing control of things such as central heating and burglar alarms, or even logging and timing phone calls.

If anyone would like to contact me they can do so on CHARD (04606) 4511.

I would be willing to write more on connecting/using modems or accessing Bulletin Boards or Retrieval Systems.

Yours sincerely,

Alan Davey
88 Halcombe Estate
Chard
Somerset
TA20 2DU

UPDATE: Since providing me with this information, Alan has almost completed his Bulletin Board, and intends to run it on Sundays from 10 am to 10 pm. He needs some indication as to the demand for such a service, so please ring or write to him to let him know your interest.

NB The "h" has been used in place of the ")"

MURPHY'S LAW No.516 Part 3 "Disk Bit Map" By Bruce Caron

The disk bit map is located on "Sector 0", which is the first sector on every diskette. Along with the bit map, sector 0, contains a lot more information about the diskette. This info is needed by the disk controller so it can read and write any data to the diskette. Sector 0 contains the following info:

Information	Size in Bytes	Sector Offset Address
Disk Name	10	h0000
No. of sectors	2	h000A
No. of sectors per track	1	h000C
Rom info, letters "DSK"	3	h000D
Proprietary disk flag	1	h0010
No. of tracks per disk	1	h0011
No. of sides per disk	1	h0012
Density per side	1	h0013
Not used	36	h0014
Disk Bit Map	180	h0038

h0000 Disk Name: The diskname can be any valid ascii code however the TI Disk manager program will not accept or display lower case characters because the manager uses the space where the character patterns are stored as a buffer area.

h000A No. Sectors on Disk: For the TI-99/4A there are only 3 values that can be stored at this location. They are:
h0168 = 360, h02D0 = 720, h05A0 = 1440 Sectors.

h000C No. Sectors per track: For a Single-Density diskette the value will be h09 = 9 sectors per track, for a Double-Density diskette the value will be h12 = 18 sectors per track.

h000D Rom info, letters "DSK": These letters are used by the disk controller to access the routines that are located in the Disk controller Rom.

h0010 Proprietary disk protection flag: This flag is used by the disk manager module to prevent copying of programs and files from a protected disk. The value for a disk that is not protected is h20 = Space character, for a disk that is protected the value is h50 = to the letter "P". Some programs may check this flag as part of a protection scheme.

h0011 No. of sectors per disk: The only two values that will be in this location will be h28 = 40 tracks per side, or some of the older drives that were made years ago were only able to step to 35 tracks. In that case the value would be h23 = 35.

h0012 No. of Sides per Disk: For Single-sided diskette the value is h01 = 1, a double-sided diskette will have the value h02 = 2.

h0013 Density per Side: A single-Density diskette has the value h01 = 1, and a double-Density diskette has a value of h02 = 2.

h0014 The next 36 bytes starting at this address are not used.

h0038 Disk Bit Map: In order match a location on the Bit Map to a physical sector on a diskette, a little numeric manipulation is required, and maybe a little background info as well.

In order for the disk controller to identify which sectors on the disk are currently being used it must have some type of look-up table to refer to. Since a byte value contains 8 Bits made of 1's and 0's, its possible to Map out 8 disk sectors with every byte in the Bit map. A "1" means the sector is used and a "0" means the sector is free.

If you take a freshly initialized diskette and examine the Byte value at Sector 0, address h0038 you will find the value h03. Remember I said that a byte value can be used to map out 8 sectors, well this is how it looks.

h03 hex = 0000 0011 binary .. This corresponds to the first 8 sectors on the diskette. If we expand this out and match the bits to the sectors we find that sectors 0 and 1 are used, and that sectors 2, 3, 4, 5, 6 and 7 are free.

Address Value	sector	7	6	5	4	3	2	1	0
h0038 = h03		0	0	0	0	0	0	1	1
		F	E	D	C	B	A	9	8
h0039 = h00		0	0	0	0	0	0	0	0
		17	16	15	14	13	12	11	10
h003A = h00		0	0	0	0	0	0	0	0

To mathematically explain a routine to calculate a bit value and position to a physical sector would take up over 3 pages of text, with many calculations in HEX and DECIMAL arithmetic. What I have done is included a table that I drew up to represent the Bit map for a Single-sided Single-Density diskette. For those that wish to expand the table, just carry on where I left off.

For some simple arithmetic we can find out how many bytes are used for each disk format.

SSSD Diskette:	360/8 =	45 Byte Bit Map
SSDD Diskette:	720/8 =	90 Byte Bit Map
DSSD Diskette:	720/8 =	90 Byte Bit Map
DSDD Diskette:	1440/8 =	180 Byte Bit Map

Bit Map Address	Length	Format
h0038 to h0064	45	SSSD
h0038 to h0091	90	SSDD
h0038 to h0091	90	DSSD
h0038 to h00D7	180	DSDD

h00D8 The remaining bytes starting at this location are not used by the disk controller and normally contain the value hFF, however some programs on disk may contain ascii characters in this location. The characters that are here are not copied by the disk manager and usually contain some reference to the software program that is on the disk.

Disk Bit Map Notes:

If you find a disk that cannot be catalogued or copied by the Disk manager program, its most probably because something on this sector is not Kosher. Go through the Bit Map info and make sure the values are correct. If the values don't match what I have given you then it doesn't belong there.

Some protection schemes actually check to see that the Bit Map is screwed up, so you may have to return things back to how you found them.

If you try to read Sector 0 and find that Disk Fixer returns an error, then the diskette is either formatted to double-density and your controller doesn't work double-density or the diskette is physically damaged. You can verify a damaged diskette by attempting to read any other sector on that particular track.

Changing any of the values in the Bit Map (h0038) should only be done if you are absolutely sure of what you are doing. This area is only used to determine how many sectors are free for a catalogue function and for a disk write operation. If you can't read a sector then changing this area won't help you.

The next page contains a screen dump of two different Disk Bit Maps. See if you can identify the following information on each.

Example No. 1
Disk name_____
Format _____
Used/Free_____

Example No. 2
Disk name_____
Format _____
Used/Free_____

```
NAVARONE IND. *** DISK FIXER V1.0 ** SECTOR DUMP    SECTOR ADDRESS  0000
ADDR =  0 1  2 3  4 5  6 7  8 9  A B  C D  E F  INTERPRETED

-------------------------------------------------------------------
0000 = 4544 2F41 534D 2020 2020 0168 0944 534B  ED/ASM    *h*DSK
0010 = 2028 0101 0000 0000 0000 0000 0000 0000   (**************
0020 = 0000 0000 0000 0000 0000 0000 0000 0000  ****************
0030 = 0000 0000 0000 0000 FF0F 0000 FCFF FFFF  ************;***
0040 = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
0050 = FFFF FFFF FFFF FFFF FFFF FFFF FF1F 0000  ****************
0060 = 0000 0000 00FF FFFF FFFF FFFF FFFF FFFF  ****************
0070 = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
0080 = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
0090 = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
00A0 = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
00B0 = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
00C0 = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
00D0 = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
00E0 = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
00F0 = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
```

```
NAVARONE IND. *** DISK FIXER V1.0 ** SECTOR DUMP    SECTOR ADDRESS  0000
ADDR =  0 1  2 3  4 5  6 7  8 9  A B  C D  E F  INTERPRETED

-------------------------------------------------------------------
0000 = 4242 532D 4E4F 562D 3834 02D0 0944 534B  BBS-NOV-84*P*DSK
0010 = 2028 0201 0000 0000 0000 0000 0000 0000   (**************
0020 = 0000 0000 0000 0000 0000 0000 0000 0000  ****************
0030 = 0000 0000 0000 0000 FF7F 0000 FCFF FFFF  ************;***
0040 = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
0050 = FFFF FF00 0000 0000 0000 0000 0000 0000  ****************
0060 = 0000 0000 0000 0000 0000 0000 0000 0000  ****************
0070 = 0000 0000 0000 0000 0000 0000 0000 0000  ****************
0080 = 0000 0000 0000 0000 0000 0000 0000 0000  ****************
0090 = 0000 FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
00A0 = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
00B0 = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
00C0 = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
00D0 = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
00E0 = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
00F0 = FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF  ****************
```

DISK BIT MAP (SECTOR 0) FOR SSSD DISKETTE

BM BYTE	SECTOR ADDRESS	SECTORS							
		007	006	005	004	003	002	001	000
1	0038	007	006	005	004	003	002	001	000
2	0039	00F	00E	00D	00C	00B	00A	009	008
3	003A	017	016	015	014	013	012	011	010
4	003B	01F	01E	01D	01C	01B	01A	019	018
5	003C	027	026	025	024	023	022	021	020
6	003D	02F	02E	02D	02C	02B	02A	029	028
7	003E	037	036	035	034	033	032	031	030
8	003F	03F	03E	03D	03C	03B	03A	039	038
9	0040	047	046	045	044	043	042	041	040
10	0041	04F	04E	04D	04C	04B	04A	049	048
11	0042	057	056	055	054	053	052	051	050
12	0043	05F	05E	05D	05C	05B	05A	059	058
13	0044	067	066	065	064	063	062	061	060
14	0045	06F	06E	06D	06C	06B	06A	069	068
15	0046	077	076	075	074	073	072	071	070
16	0047	07F	07E	07D	07C	07B	07A	079	078
17	0048	087	086	085	084	083	082	081	080
18	0049	08F	08E	08D	08C	08B	08A	089	088
19	004A	097	096	095	094	093	092	091	090
20	004B	09F	09E	09D	09C	09B	09A	099	098
21	004C	0A7	0A6	0A5	0A4	0A3	0A2	0A1	0A0
22	004D	0AF	0AE	0AD	0AC	0AB	0AA	0A9	0A8
23	004E	0B7	0B6	0B5	0B4	0B3	0B2	0B1	0B0
24	004F	0BF	0BE	0BD	0BC	0BB	0BA	0B9	0B8
25	0050	0C7	0C6	0C5	0C4	0C3	0C2	0C1	0C0
26	0051	0CF	0CE	0CD	0CC	0CB	0CA	0C9	0C8
27	0052	0D7	0D6	0D5	0D4	0D3	0D2	0D1	0D0
28	0053	0DF	0DE	0DD	0DC	0DB	0DA	0D9	0D8
29	0054	0E7	0E6	0E5	0E4	0E3	0E2	0E1	0E0
30	0055	0EF	0EE	0ED	0EC	0EB	0EA	0E9	0E8
31	0056	0F7	0F6	0F5	0F4	0F3	0F2	0F1	0F0
32	0057	0FF	0FE	0FD	0FC	0FB	0FA	0F9	0F8
33	0058	107	106	105	104	103	102	101	100
34	0059	10F	10E	10D	10C	10B	10A	109	108
35	005A	117	116	115	114	113	112	111	110
36	005B	11F	11E	11D	11C	11B	11A	119	118
37	005C	127	126	125	124	123	122	121	120
38	005D	12F	12E	12D	12C	12B	12A	129	128
39	005E	137	136	135	134	133	132	131	130
40	005F	13F	13E	13D	13C	13B	13A	139	138
41	0060	147	146	145	144	143	142	141	140
42	0061	14F	14E	14D	14C	14B	14A	149	148
43	0062	157	156	155	154	153	152	151	150
44	0063	15F	15E	15D	15C	15B	15A	159	158
45	0064	167	166	165	164	163	162	161	160

E N H A N C E D B A S I C

An enhancement of TI BASIC available through STATISTICS and PRK modules

Peter Brooks February 1985

References: TI Document ARCHIV.PRK.DOC.SUBRLS1 courtesy of TI
Articles by, and personal communication with, PAUL W. KARIS

CALL G() - THE GETPUT SUBPROGRAM : I THOUGHT WE'D NEVER GET HERE...

Well, in fact we haven't, yet. If you thought you'd seen the last of CALL H(), you'd be only half-right. However, there is one aspect of CALL A(), the Accept subprogram, which has a bearing on the skeleton structure of our database implemented through CALL H().

One form of CALL A() is:

```
CALL A(Y,X,W,C,V,F)
```

or

```
CALL A(Y,X,W,C,V$,F)
```

The first five parameters we have already dealt with - Y and X, the row and column co-ordinates; W the width indicator; and C the return code which indicates which 'entry' key was used. V or V$ are the numeric and string variables which will be the destination of any data which is entered.

We have not discussed F before. I decided that we would look at it in detail later in the series. Now seems a suitable point at which to include it.

The F parameter is only valid when a completely-defined file header exists. That is, until you have used CALL H() to define the structure and detailed specification of the "form", you can't make use of the F parameter in CALL A().

F refers to the FIELD NUMBER (FLD) which we encountered in the discussion on CALL H(). In our CLUB-85A file, FLD number 1 is TITL-INIT (see V1.11 TI-LINES), FLD 2 is DEAR, FLD 3 is SURNAME, and so on. What F does in CALL A() is to perform a comprehensive input validation on the data being entered, by fetching the appropriate characteristics (type, width, number of decimal places, etc.) of the FLD specified by F, and comparing them with the characteristics of the entry being attempted.

TITL-INIT's characteristics are: CHAR, Width = 15, so any entry which is not within these limits is rejected with a 'honk'. (Honk is the term used to describe the error tone. The 'accepted' tone is called a beep.)

Check the summary tables in TI-LINES V1.11, especially page 18.

This facility saves you some of the work involved in writing your own routines to validate any input, but of course it only applies to the use of CALL A().

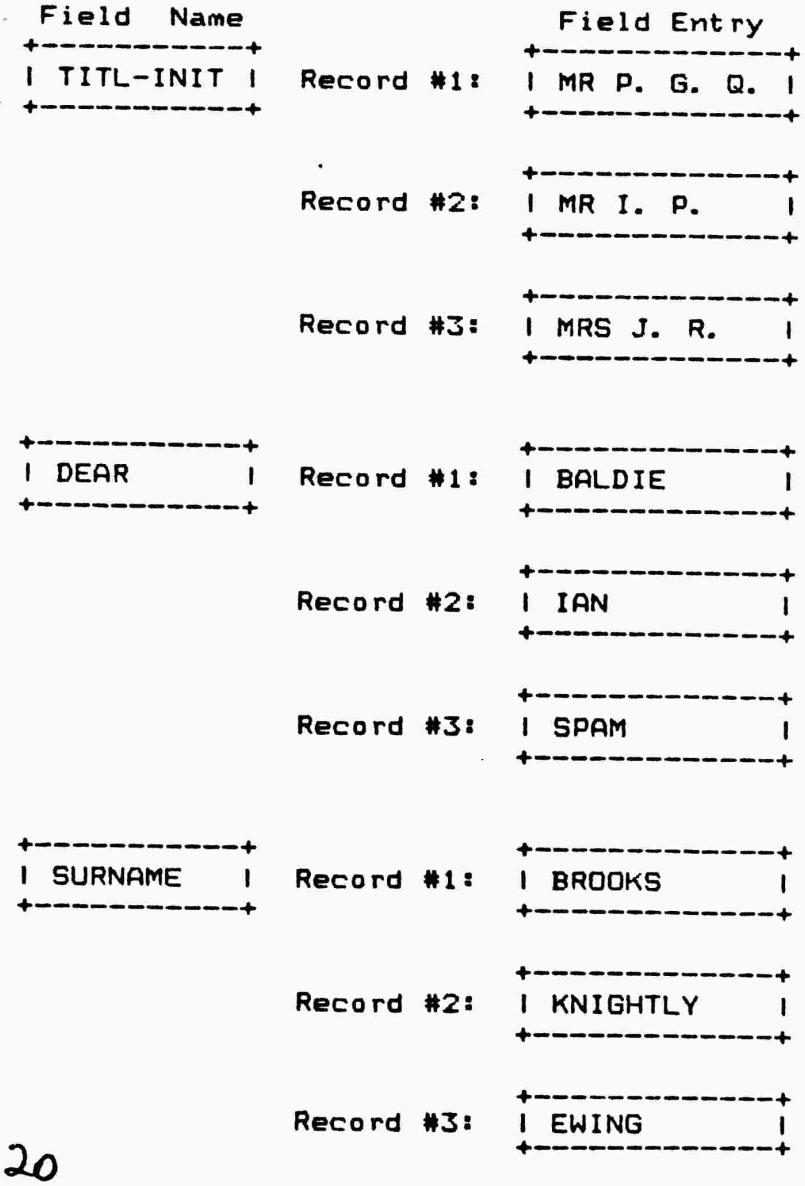
And now...

CALL G()

Having created the skeleton structure (the Header) of our "form", we now need a means of filling the form with data. Each completed form ("page" or record) will be a component in our club file (Database).

CALL G(), the GETPUT subprogram, gives us the ability not only to write on any form but also to read entries made on any existing form. GETPUT addresses itself to the fields in each record - in our case, the TITL-INIT, DEAR, SURNAME, etc., fields.

It is important to make one thing clear here. A field has two parts - a "label", or "prompt" if you like, (e.g. TITL-INIT, DEAR, SURNAME) and an entry (MR P. G. Q., BALDIE, BROOKS). You used CALL H() to put the label into the Header (the description of the layout), and CALL G() will be used to make entries - or examine them.



The item under Field Name is fixed or constant - it never changes from record to record. The items under Field Entry usually change from record to record - unless there are more than one entry with the same name, for example.

We come now to the different forms which CALL G() can take, and although TI's own document is published elsewhere in this issue courtesy of TI, it is perhaps best to ignore it as it is very misleading.

The variants are:

1. CALL G(W, REC, FLD, V or V$)

2. CALL G(R, REC, FLD, MIS, V or V$)

R and W are Read and Write codes as before, although they are slightly different in this case. Note that there are two distinctly different forms: one for Reading, and one for Writing. The codes are different too:

0 = write valid data to the file (i.e. W)

1 = read data from the file (i.e. R)

2 = indicate missing data in the file

This last code concerning "missing data" might cause a little confusion. Essentially it means a field in which no entry has been made (i.e., that field on the form has not been filled in).

Note that the (1) form of CALL G() is used only when writing, and the (2) and (3) forms are used only when reading.

As an aside, I find it a little daft that zero should be used as the code for writing data to a file. If by chance an error has been made in the listing of a program utilising such subprograms, and the wrong variable is used to indicate a Write when in fact a Read is required, an overwrite of at least one field could occur which might not be noticed until it was too late. As zero is the default value of all unassigned numeric variables it would have been sensible to use a non-zero value to indicate a Write, and zero to indicate Read.

When reading from a file with R set to 2 (indicate missing data) a value will be returned in MIS according to what is found by CALL G().

If MIS = 0, data has been found, while of MIS = 1, this indicates that a "null entry" has been found; i.e., a field entry with nothing in it, neither numbers nor characters.

If missing data is found (which sounds like a contradiction in terms) one other effect occurs: the contents of the "return variable" (V or V$) are left unchanged. This permits you to make use of "defaults", but it might cause problems under certain circumstances.

To make an entry in our CLUB-85A file therefore, we need to refer to the structure which we set up using CALL H() back in issue V1.11. The tables on pages 17 and 18 of that issue give details of the kind of entry needed in each case.

(Bear in mind that the use of "lower case" - the small capitals - may lead to unusual effects if the file is ever examined through the normal use of PRK or Stats, because they are not defined to be alphanumeric characters by those modules).

FLD	ITEM	EXAMPLE ENTRY	COMMENT
1	TITL-INIT	MR P. G. Q.	
2	DEAR	BALDIE	
3	SURNAME	BROOKS	
4	AD1	61 THE AVENUE	
5	AD2	KENNINGTON	
6	AD3	OXFORD	
7	AD4		MISSING
8	AD5		MISSING
9	POST-CODE	OX1 5PP	
10	PHONE	0865 730044	
11	SEX	M	
12	MEM-TYPE	L	
13	RENEWAL	31DEC99	
14	SUM-PAID	0	

Our first example entry is based upon my details. We can place it in memory using an approach similar to that used with CALL H() previously.

We need to place the structure of the database in memory, so our first action must be to start from page 18's NITTY GRITTY (V1.11) - unless of course you did use CALL S() as instructed and have already stored the "header". If you did so, then load the file back in thus:

Switch the computer on and insert the PRK (or Stats) module.

Select TI BASIC and type:

```
CALL P(10000)
```

(and then press ENTER)

followed by:

```
NEW
```

(and ENTER).

Once again we will need to make use of a command which has not yet been discussed. Type either:

CALL L("CS1",A)

or

CALL L("DSK1.CLUB-85A",A)

depending on whether you used cassette or disk storage, and on the file name you chose (where applicable).

Using PRINT A afterwards will indicate whether the transfer of data was successful, as for CALL S() in V1.11.

We will discuss CALLs L and S next time.

Having placed the structure in memory, we can begin to fill in the "form" with our example entries.

First, though, you might like to confirm that the structure is the one you want:

Type:

CALL H(1,1,0,V$)

and press ENTER.

This breaks down to be an instruction to:

CALL H(READ, INFO ITEM 1 - the file name - see p16 V1.11, FIELD NUMBER - unimportant therefore ZERO, AND PUT IT IN V$)

PRINT V$

and you should see "CLUB-85A" - or whatever you stored there originally.

(If anyone would like a follow-up to demonstrate how to read a structure item by item, I would be happy to oblige).

On to the first entry.

The general form of CALL G() which we will be using will be:

CALL G(0,REC,FLD,V or V$)

which translates as:

CALL G(WRITE, RECORD or PAGE number, FIELD number, NUMBER or NUMERIC VARIABLE or STRING or STRING VARIABLE)

We can either place each item individually using the Immediate mode, or write a short routine to do it for us. In order that I may give brief explanations where necessary, I will only demonstrate CALL G() here in the Immediate mode.

There are 14 fields to be entered. Each CALL will have the same initial part:

```
CALL G(0,1,
```

that is:

```
CALL G(WRITE, RECORD #1,
```

and knowing this should help to minimise typing errors.

Let's begin.

To enter the TITL-INIT item (field 1):

```
CALL G(0,1,1,"MR P. G. Q.")
```

Then to check that it was accepted:

```
CALL G(2,1,1,MIS,V$)
```

and `PRINT MIS,V$`

which should result in:

```
0          MR P. G. Q.
```

If you see anything else, there is something seriously wrong. See me afterwards!

To continue:

```
CALL G(0,1,2,"BALDIE")
CALL G(0,1,3,"BROOKS")
CALL G(0,1,4,"61 THE AVENUE")
CALL G(0,1,5,"KENNINGTON")
CALL G(0,1,6,"OXFORD")
CALL G(0,1,7,"")
CALL G(0,1,8,"")
```

NB These last two are null strings - two quotes together.

```
CALL G(0,1,9,"OX1 5PP")
CALL G(0,1,10,"0865 730044")
CALL G(0,1,11,"M")
CALL G(0,1,12,"L")
CALL G(0,1,13,"31DEC99")
CALL G(0,1,14,0)
```

Note the last entry - it is the only one which is a "non-character" type.

Now to demonstrate the effects of a READ code of 2 (indicate missing data). Two fields (7 & 8) had no entries, so we'll use on of them.
=one

Theoretically,

```
V$ = "BEFORE"
CALL G(1,1,7,MIS,V$)
PRINT MIS,V$
```
* [should be CALL G(2,1,7,MIS,V$)]

should produce a display of:

```
      1         BEFORE
```

indicating that the data is "missing" (1) and that the contents of V$ ("BEFORE") are unchanged.

Use a field value of 9 (POST-CODE):

```
      V$ = "BEFORE"
      CALL G(1,1,9,MIS,V$)     [Correction: should read CALL G(2,1,       ]
      PRINT MIS,V$
```

and you should see:

```
      0         OX1 5PP
```

At a later stage I will print out a file as it appears when stored on disk, which, while of little real interest to disk non-owners, should at least give them something more concrete to work on.

Next issue we will round off with a look at the two CALLs L and S, which we have used already. This should then be followed by two or three presentations from PAUL KARIS and FRANC GROOTJEN of the NETHERLANDS TI USERS, and there will be room for discussion and debate in subsequent issues for those with a burning interest (no, not pyromaniacs!).

It is worth bearing in mind the information presented elsewhere in this issue by JEREMY BYGOTT, who has noted certain effects which can occur under some circumstances and which could lead to a loss of data. Until we can identify and correct the fatal errors upon which Jeremy comments, I would not advise you to commit valuable data to any database created under whatever circumstances with either the PRK or Stats modules.

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

TI DISK OPERATING SYSTEM (DOS)

BY DENNIS HANCOCK
COMPILED FROM "THE SOURCE"

REPRODUCED FROM THE SASKATOON TI COMPUTER CLUB NEWSLETTER
WITH PERMISSION

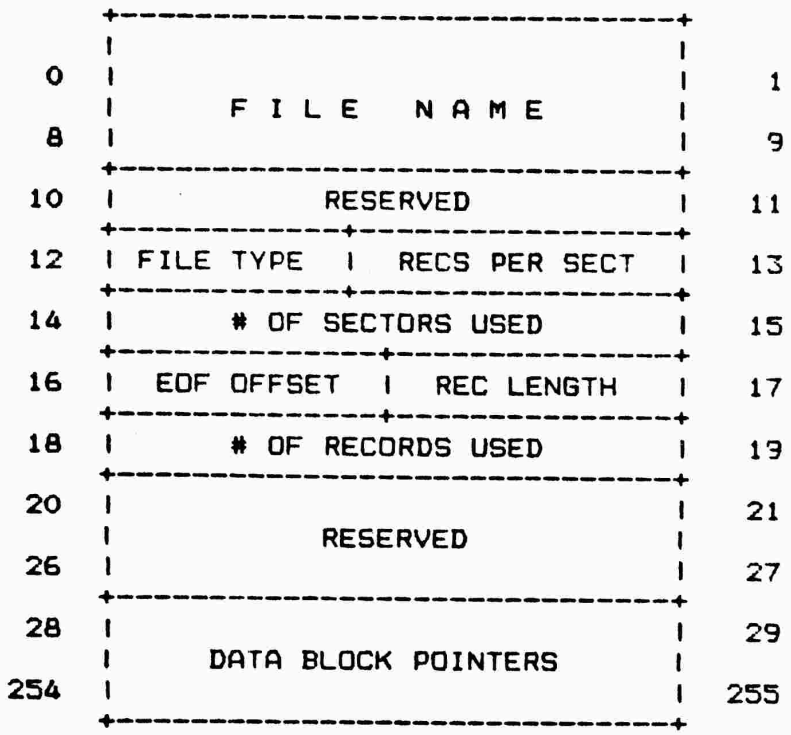
THE FILE DESCRIPTOR BLOCK

In the last issue we learned the layout of sector 0, the Volume Information Block (VIB). This first sector holds info on the disk status including diskname, # of sides, track density, and the # of free tracks on the disk. Lastly, sector 0 contains the infamous "disk bit map" which indicates exactly which sectors are entered in the order they are created (not alphabetically!).

1. The FDRs are scanned, sorted and then their sector numbers are reprinted onto sector 1 in the NEW alphabetical order. This indexing helps to speed up file access and cut down on wasted disk space.

WHAT DOES IT LOOK LIKE?

Below is a diagram of the FDR. As you can see, it looks much like the VIB. We will describe each of the sections later in the article.



FDR DESCRIPTION

Bytes 0-9 contain the filename (up to ten ASCII characters - padded if necessary).

Bytes 10-11 are reserved for future expansion.

Byte 12 contains the file type flag. The bits are set according to the file attributes in TI-BASIC (Internal, Display, Fixed, Variable, etc.) and can be interpreted as follows:

BIT	MEANING
0	0 = Data file 1 = Program file
1	0 = DISPLAY format 1 = INTERNAL format
2	RESERVED
3	0 = Unprotected file 1 = Protected file
4-6	RESERVED
7	0 = Fixed length recs 1 = Variable length recs

For example: if byte 12 contained >80 (b10000000) you'd know it was a Display/Variable data file.

Byte 13 contains the number of records per sector. For example, if the file was a DIS/VAR 80 file then byte 13 would contain >03 (3=240). Note that TI-DOS automatically takes care of any "blocking factors" that may be needed. TI-DOS accesses the disk in 256 byte blocks. This means it does not "split" any records between sectors. In other words, any record more than 128 bytes long takes up an entire sector for storage! Keep that in mind next time you plan your data files!

Bytes 14-15 contain the number of sectors used by the file.

Byte 16 contains the EOF Offset for the last sector in the file. Since the DOS accesses in 256 byte blocks, this value is used to locate the last byte in the file. This prevents reading past the end of the file. This is only used for variable length data files and for program files.

Byte 17 contains the record length. If the file is 80 bytes long, byte 17 contains >50. If the file is variable in length, this value is the maximum length allowed.

Bytes 18-19 contain the number of records allocated for the file. This is either the number of records presently on file or the number of records the file was initially "opened for" in the TI-BASIC OPEN statement. If the file is VARIABLE-type, this value is the same as the value in bytes 14-15, but in REVERSE ORDER!

Bytes 20-27 are RESERVED and set to 0.

Bytes 28-255 contain the data pointers. When the file must be "broken up" due to its size, a reference to the next record of the file is entered in the pointer area. This tells TI-DOS where on the disk to find the next block of records for this file. Each data chain pointer consists of two THREE byte entries. The first entry contains the sector number of the START of the new data block. The second entry contains the "EOF offset" of THAT block (not necessarily the EOF of the FILE!) To make matters worse, the three bytes are stored in a rather awkward manner. See the diagram below:

```
Start Sector :  S3 S2 S1
Block "EOF"  :  B3 B2 B1
```

Note that the bytes are stored in "reverse" order or right to left.

Now the two sets of three bytes are stored in a 6 byte segment as follows:

```
S2 S1 B1 S3 B3 B2
```

Note the location of bytes S3 and B1!

As each new block is created, a six-byte entry is added to the data chain pointer area. The pointer area can handle up to 76 different blocks for the same file.

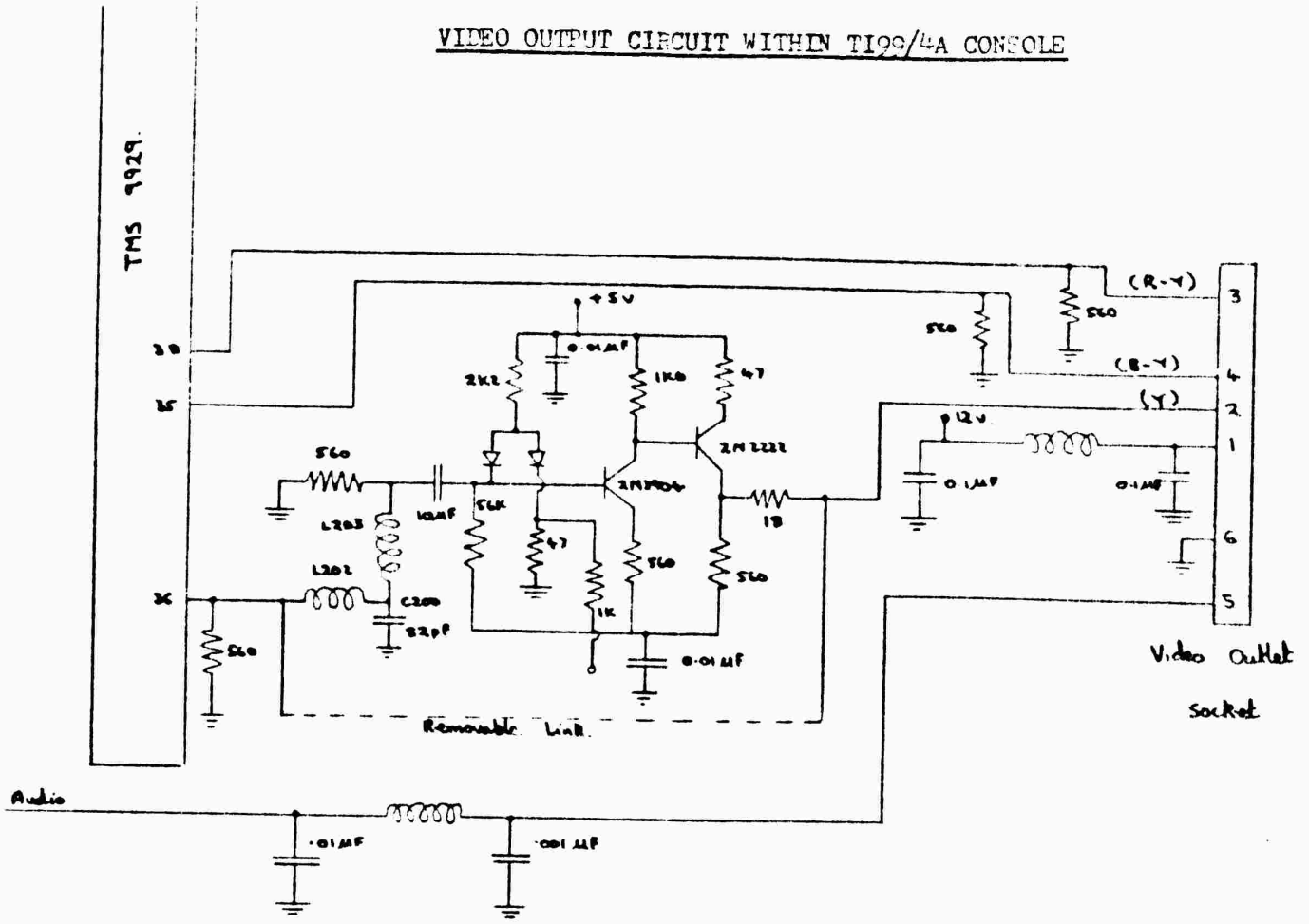
L E T T I N G I T S I N K I N

Now get out your disk-reading program and start looking at the FDRs. What are the file attributes ? Experiment with changing the protection bit, or the record length and see what happens. Can you figure out a way to read a program file like a data file ? See if your data files have any data chain pointers. If so, follow them down and find the next starting sector. After you get comfortable with the information contained in the FDRs you will be ready to do a little file-handling of your own including recovering already deleted files, and storing "blown directories". You will even be able to follow them down and find the next starting sector. You will be able to retrieve those DIS/FIX 0 files you ended up with when you downloaded from SF99 with TE-II and forgot to close the file!

RGB AND YOUR TI-99/4A

By VIV COMLEY

VIDEO OUTPUT CIRCUIT WITHIN TI99/4A CONSOLE

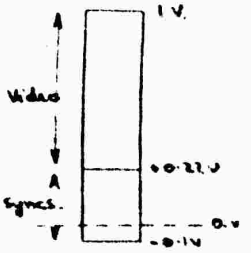


R = Red B = Blue Y = Luminance

Luminance (Y) Signal

The luminance signal consists of the 'brightness' information and the synchronization pulses.

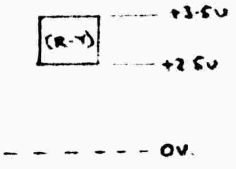
The measured luminance signal appears as sketched.



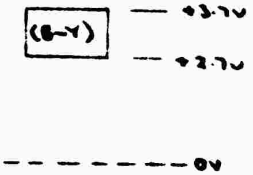
Red minus Luminance (R-Y) Signal

This signal contains a portion of the colour information.

The measured (R-Y) signal is;
(There are no synchronization pulses on this signal).



Blue minus Luminance (B-Y) Signal



This signal forms the remaining part of the colour information. It also contains no synchronization pulses.

The measured signal levels are;

Audio

The sound information comes at around 0.6 volts peak to peak.

All the above signal voltages were measured on one particular TI99/4A console only, but there is no reason to suspect other TI99/4A's should not produce similar voltage levels.

Discussion

The colour television picture is built up from the three colour information signals Red, Green and Blue.

However, for various reasons (i.e. compatability between colour and monochrome receivers, so that a monochrome T.V. will display a colour picture correctly in black and white, and other reasons), the colour television signal is formed from the three components,

1 Luminance (Y) information, or just the shades of grey of a picture.
2 Red minus luminance (R-Y) information
3 Blue minus luminance (B-Y) information

Because R+G+B=Y, (with suitable weighting parameters in practice), then the green (G) information may be reconstituted at the receiver by suitable additions (in the RGB matrix) of the Y, R-Y, and B-Y signals.

The TI99/4A console does not directly produce the R, G, and B information (or at least it is not available to the user), but it feeds the external PAL coder and modulator with the (R-Y), (B-Y) and Y signals at the voltage levels shown earlier.

The PAL coder uses the (R-Y) and (B-Y) signals to phase modulate a high frequency signal (known as the colour subcarrier), which is then superimposed upon the Luminance (Y) signal. This combination signal, along with the audio information is translated up in frequency by the modulator to channel 38.

The T.V. receiver has to receive this signal, filter and amplify it as it does any 'off-air' T.V. signal, and then decode the (R-Y), (B-Y) information. These signals are then suitably combined with the Y signal to finally produce the Red, Green and Blue information to drive the picture tube. As a sideline to this process, the receiver must also extract the synchronization information to control its line and field timebases, and the audio information.

A number of home computers have available the Red, Green and Blue signals directly, and these may be fed to a suitable RGB monitor.

An important point to note here is that there are two ways of producing this RGB signal.

1. The 'easy' way, as accomplished by the BBC micro among others. I say the 'easy' way but in some ways it is very convenient and has advantages. As the computer produces logic level voltages (i.e. 0 and 1) it would appear to make sense to produce the colour picture simply by making each Red, Green and Blue signal 0 or 1 as necessary. With this system we can produce eight colours.

   Red only
   Green only
   Blue only
   Red + Green = Yellow
   Red + Blue = Magenta
   Green + Blue = Cyan
   Red + Green + Blue = White
   No signal (all 0's) = Black

   Acorn claim sixteen colours for the BBC micro, but the other eight I believe are produced only by flashing the first set of eight. i.e. Red flashing on and off
   Blue flashing on and off
   etc.
   A bit of a swiz really!

   However these logic levels can use fairly standard and cheap (and fast) TTL (Transistor - Transistor Logic) circuitry, and RGB monitors for use with this type of signal have what are known as TTL inputs.

2. The 'hard' way. In this system the colour T.V. signal is an analogue signal. i.e. it can take any value between two levels and not just 0 or 1. (i.e. 0, 0.1, 0.2, 0.3 ---- 1.0). With this system we can produce a full sixteen colour (or more) set, and include levels such as dark green, light green, etc.

   The reason why this is the hard way, though it gives more control over the colour set available, is that all circuitry must be analogue circuitry, with all the problems that involves. Such as linearity, fast response times, etc.

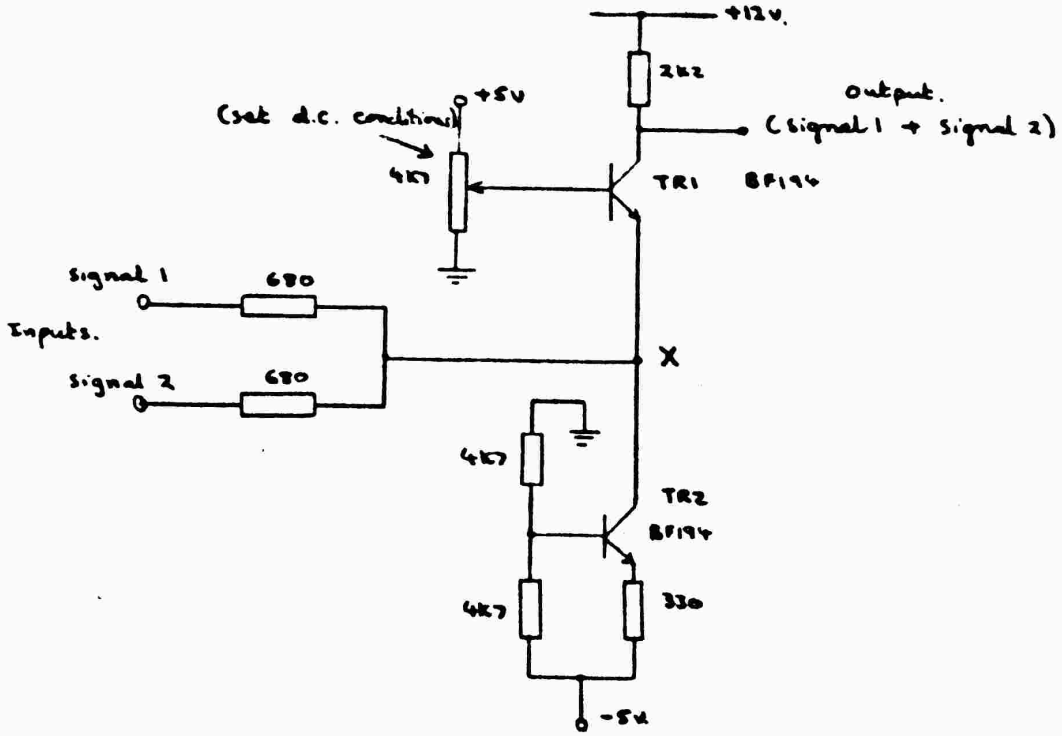
   Needless to say the TI99/4A uses this system, but as a result gives us a true sixteen colour set.

The RGB matrix described here was designed to operate with a microvitec monitor, requiring the following inputs.

Red Green Blue Synchronization pulses	All at a voltage level of 1 volt peak to peak into a load of 75Ω

The circuitry is fairly straightforward in operation, although at first sight it may seem complex.

A basic part of the circuit is the following summing circuit, formed from two transistors, and capable of summing two signals together. (i.e. (R-Y) + Y = Red. etc.)



The summation of two voltages is accomplished at a low impedance (or 'virtual earth' point), so to prevent one input from 'seeing' what the other input is doing.

The above circuit achieves this virtual earth as follows. (Note that I claim no originality for this circuit, which is fairly standard). Transistor TR2 forms a constant current emitter load for transistor TR1. The voltage on the base of TR1 is set and maintained (essentially) constant by the 4K7 variable resistor. TR1 will pass just enough current to ensure that its emitter voltage is equal to the base voltage (less one diode drop in practice). Any attempt by the input signals to feed current into point X and so change TR1 emitter voltage will be counteracted by TR1 adjusting its current.

Remember that TR1 will attempt to maintain the voltage at point X Constant. These current changes are reflected by a change in the voltage across the 2K2 (2,200 ohm) resistor in TR1's collector. These voltage changes form the output signal.

Two points of interest to note here.

1. We cannot really use operational amplifiers to do the summing, as slow-rate limitations will occur unless we use very expensive op-amps. Discrete components can do the job very well.

2. We cannot use capacitive coupling within the circuit as it is important to maintain the d.c. component of the signal. If we do use capacitive coupling we have to d.c. restore somehow. It is best to use d.c. coupling throughout the circuit, which is done.

The circuit for the RGB matrix is attached. The circuit has excellent high frequency performance, which causes a problem in that instability (coupling at high frequencies occurring between input and output) is a problem, and the final circuit was constructed on double sided printed circuit board, with one side used as an earth plain.

The power supply is not shown, as this is fairly standard. Integrated circuit regulators are required in the power supply with the +5 volt and -5 volt regulators being rated at around 1 amp.

The synchronization pulse separator removes the synchronization pulses on the luminance (Y) signal, and makes these available as a separate output.

The audio amplifier was constructed from bits and pieces in the spares box. It by no means represents state of the art; the OC71 and OC72 transistors being almost historical curiosities! It is included for completness however.

The unit was constructed in a metal case, and makes a reasonably compact and respectable looking job.

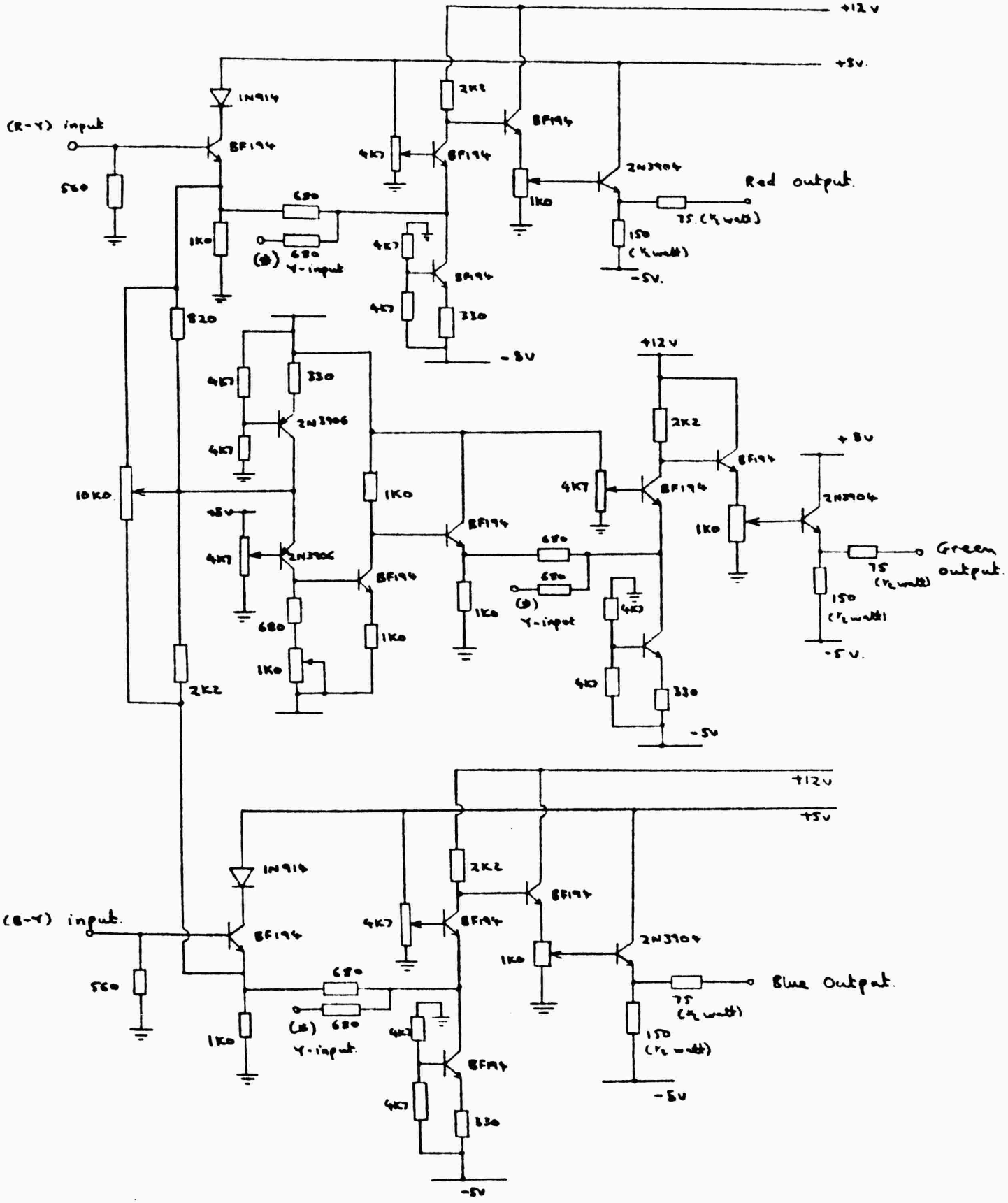
When setting the circuit up, it is essential to have an oscilloscope available, and a knowledge of what to expect from video signals. In setting up I found the master title screen produced by the TI99/4A (the identification screen with the colour step wedge) very useful.

A few diodes around the input stages of the matrix are used to prevent damage to the TI99/4A console should the console be switched on with the RGB matrix unpowered.

A useful reference book for the matrixing of the signals to produce R, G and B information from the (R-Y), (B-Y) and Y signals, may be found among those dealing with PAL colour television systems. e.g. "Principles of PAL colour television" by Simms.

Please note that the output from the RGB matrix is not suitable for feeding to a monitor with TTL inputs. It will only supply an RGB monitor requiring red, green and blue signals at 1 volt peak to peak into 75Ω and synchronization pulses. (These are standard video levels).

RGB matrix. (Y amplifier on separate sheet)

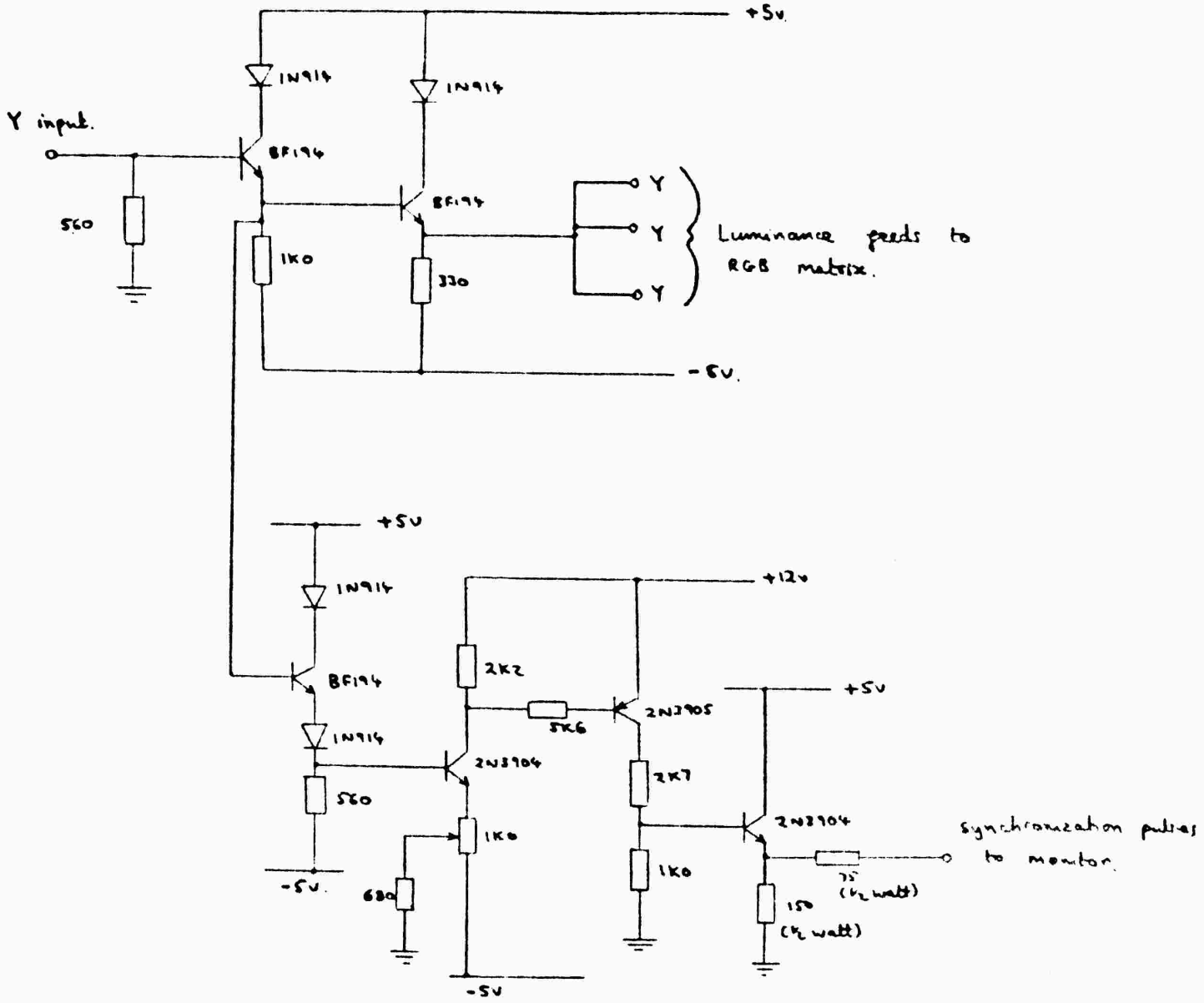


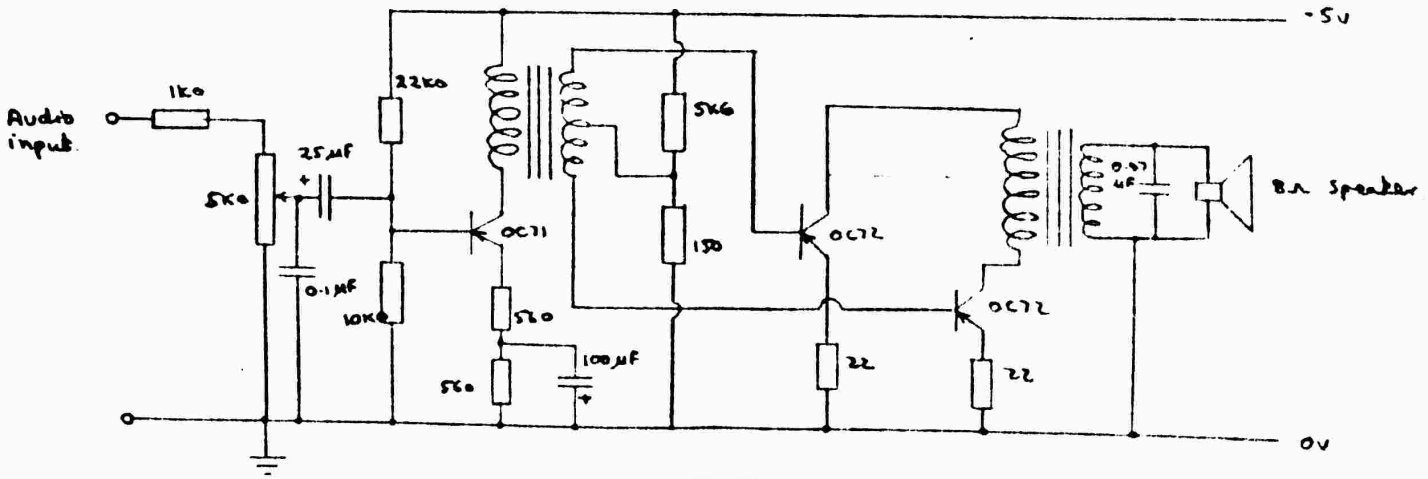
All resistors are ¼ watt unless stated. Tolerance 10%

Terminals marked (*) are luminance input from luminance amplifier.

All resistor values are in ohms. e.g. 560 = 560 Ω. 2K2 = 2,200 Ω.

Synchronization separator, luminance amplifier and audio amplifier





GENERAL DESCRIPTION OF THE GETPUT SUBPROGRAM

The GETPUT subprogram is a subroutine resident in the Personal Record Keeping and Statistics command modules. When the command module is plugged in, this routine can be called from a BASIC program . It is used to write data to and read data from a file which has been defined using the PREP and HEADER subprograms . The CALL statement is used to execute this routine and takes one of the following forms:

1. CALL G(R/W,REC,FLD,V)
2. CALL G(R/W,REC,FLD,V$)
3. CALL 6(R/W,REC,FLD,MIS,V2)
4. CALL G(R/W,REC,FLD,MIS,V2$)

where

R/W	=	read/write code	(numeric expression)
REC	=	record number	(numeric expression)
FLD	=	field number	(numeric expression)
V	=	numeric value	(numeric expression)
V$	=	string value	(string expression)
MIS	=	return code	(numeric variable)
V2	=	return variable	(numeric variable)
V2$	=	return variable	(string variable)

The value of R/W determines whether a read or write is done. The values are:

0 = write valid data to the file

1 = read data from the file

2 = indicate missing data in file

Any other value for R/W will cause an error.

REC is the record number. In Personal Record Keeping this is referred to as page number, and in Statistics as observation number. If the value is higher than the highest possible record number, an error occurs. If a write is done to a higher record than the highest previously defined record, the higher record number is stored in the header as the new highest record number. If a read is done from an undefined record, the results are unpredictable.

Therefore, it is important to create records sequentially without skipping any. A negative number or zero also yields unpredictable results.

FLD is the field number within the specified record. In Personal Record Keeping this is referred to as item number, and in Statistics as variable number. Each record has the same defined fields. If the field number is higher than the highest defined field number an error will occur. A negative or zero value will generate unpredictable results.

V and V$ are values to be written to the file. If they are expressions, the result of evaluating the expression is stored. If the value is incompatible with the definition of the specified field, the results are unpredictable. For example, if a field is defined to be decimal type with a width of 6 and 2 decimal places, both 7654 and 7.654 are incompatible values. 7654 has 1 too many digits to the left of the decimal point, and 7.654 has 1 too many digits to the right of the decimal point. When indicating missing data in the file, V or V$ must be included in the CALL parameters. However, the value of V or V$ is ignored.

MIS is a numeric variable in which a code is returned to indicate normal versus missing data. This parameter must be included when reading from the file, but must not be included when writing. After a read MIS = 0 indicates data found, MIS = 1 indicates missing data.

V2 is a numeric variable in which the numeric value being read from the file is returned. V2$ is a string variable in which a string value is returned. If a numeric variable is used when reading a string item, or a string variable is used when reading a numeric item, an error will occur. If missing data is found by the read, the return variable is not changed.

D A V I D ' S C O L U M N

A sort of Brown Study ?

Hello again,

Welcome to the column of TI-LINES written by the younger members. (I still haven't got a name for it yet. A free badge is on its way for anyone who thinks up a good, original name.) At last we have a submission from another member of the club. It is from RICHARD OWEN of South Wales. The Extended Basic program "Doodle" which is included here is his work.

Perhaps it was the offer of a free game that encouraged him? Still, I'm not complaining - it's help. I still have two more games to offer if anyone wants to submit anything - anything about the TI will do. By the way, the two games are SNOUT OF SPOUT and INTRIGUE PENTATHLON if anyone's interested.

Here is Doodle by Richard Owen.

Doodle runs in Extended Basic, but it can be made to run in Basic if 2 lines are deleted. These are lines 300 and 310. Also lines 260-290 have to be altered slightly. They will have to read:-

```
260  IF C<2 THEN 261 ELSE 270
261  C = 2
270  IF C> 32 THEN 271 ELSE 280
271  C = 32
280  IF R> 24 THEN 281 ELSE 290
281  R = 24
290  IF R< 2 THEN 291 ELSE 320
291  R = 2
```

The changes that are listed above should only be made if you want the program to run in Basic.

The control keys are:-

0,1,R,W,P,Z,X,C,S,D,E

0 - makes the cursor black
1 - makes the cursor transparent
R - moves cursor diagonally (up and to the right)
W - moves the cursor diagonally (up and to the left)
P - saves the doodle onto tape (opens a file)
Z - moves cursor diagonally (down and to the left)
C - moves cursor diagonally (down and to the right)
S - moves cursor left
D - moves cursor right
X - moves cursor down
E - moves cursor up

That is all the information you will need to use this excellent program which is on the next page.

Good computing,

Dave Brown.

```
100 REM ********BY RICHARD OWEN********
110 REM ***WHILE GETTING BORED AT ***
120 REM ***********HOME 1985***********
130 CALL CLEAR
140 REM ********************************
150 REM ***********#DOODLE#************
160 REM ********************************
170 CALL CHAR(42,"FFFFFFFFFFFFFFFF")
180 CALL CHAR(76,"FF8181818181FF")
190 R=12
200 C=16
210 Z=42
220 CALL KEY(3,K,S)
230 IF S=0 THEN 220
240 ON ERROR 220
250 ON POS("01PWRZCSDXE ",CHR$(K),1)GOSUB 670,690,540,420,450,480,510,340,360,38
0,400,650
260 IF C<2 THEN C=2
270 IF C>32 THEN C=32
280 IF R>24 THEN R=24
290 IF R<2 THEN R=2
300 CALL DELSPRITE(#1)
310 CALL SPRITE(#1,96,5,(R*8)-8,(C*8)-8)
320 CALL HCHAR(R,C,Z)
330 GOTO 220
340 C=C-1
350 RETURN
360 C=C+1
370 RETURN
380 R=R+1
390 RETURN
400 R=R-1
410 RETURN
420 R=R-1
430 C=C-1
440 RETURN
450 R=R-1
460 C=C+1
470 RETURN
480 R=R+1
490 C=C-1
500 RETURN
510 R=R+1
520 C=C+1
530 RETURN
540 OPEN #1:"PIO.LF=0"
550 FOR A=1 TO 24
560 FOR B=1 TO 32
570 CALL GCHAR(A,B,D)
580 B$=B$&CHR$(D)
590 NEXT B
600 PRINT #1:CHR$(14);B$
601 PRINT #1:CHR$(14);B$
602 PRINT #1:CHR$(14);B$
603 PRINT #1:CHR$(14);B$
604 PRINT #1:CHR$(10)
610 B$=""
620 NEXT A
630 CLOSE #1
640 RETURN
650 CALL CLEAR
660 RETURN
670 Z=42
680 RETURN
690 Z=32
700 RETURN
```

BRIEF NOTES

HOWARD GREENBERG of ARCADE HARDWARE has dropped me a line concerning the current situation with respect to some of his products. I would also like to take this opportunity to advise software and hardware suppliers that they may advertise their products in TI-LINES without charge, as has been OTIU policy since the beginning. There are too few OTIUers for it to be a worthwhile market for which a commercial concern might consider normal advertisements. I will do my best ensure that advertising copy is reproduced as clearly and as neatly as possible

Please do take advantage of this facility.

Over to Howard:

"I've several new products either in stock or on the way, about which your readers might like to know. They aren't all going to be available from stock, or even immediately, but in every case they are at the very least on order.

The products are:

HARDWARE

Myarc RS232 card - £125.00 (from stock). Gives 19200 baud rate and TRUE Centronics output.

Myarc Disk Controller - £185.00 (Due in week from 27th March). Controls DS/DD or any lesser drive (e.g. TI format). Also provides several CALLs for BASIC, including CALL LINK and CALL LOAD. It also has a very sensible CALL DIRectory so that you don't need to keep swapping the Disk Manager module to see what's on disk.

Myarc's Mini-Box - (Due in within 21 days of 27th March) £495.00.
Reconfigured without an on-board disk drive, but with space and power for two half-heights. Includes RS232, Centronics, 32K RAM, and Disk Controller.

Myarc 128K RAM card (to order only) - £249.95. Not a true 128K at all, but 32K normal plus 96K RAMdisk/print spooler. A bit pricey, but a few have expressed interest for certain applications such as software development where disk access times are too slow.

SOFTWARE

AtariSoft are back in business with their range of games. They're now at a nice price too.

Donkey Kong, Defender, Ms Pac-man, Moon Patrol, and Pole Position are all at £14.95.

Pac-man and Jungle Hunt are both £9.95 each.

All will be available around mid- to late- April.

Extended BASIC - £74.95 - newly-manufactured with a new and bigger manual. The price is a bit hideous, but at least its there. Avilable mid- to late- April.

DFX Screen Dump - disk-based for Epson printers (and compatibles, which includes most matrix machines) - £24.95. It also requires the fitting of a LOAD INTERRUPT SWITCH. This program will dump ANYTHING, Super Sketch, modules, etc., to printer.

Disk Repair Kit - £29.95 - available now. It's written by a friend of mine and is my first attempt at software production. In every way, it is better than the Navarone Disk Fixer it replaces. (And cheaper, too!).

Infocom Adventures - £various, depending upon game. These are THE adventures, disk-based and very pricey, but you do get a great deal for the money. Titles are: ZORK I, ZORK II & ZORK III, Enchanter, Deadline, Witness, Starcross, Planetfall, Infidel, Sorcerer, Cut-Throats, and the NEW one - Hitch Hikers Guide To The Galaxy.

They should all be due in around mid- to late- April."

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

ARCADE HARDWARE

211 Horton Road Fallowfield Manchester M14 7QE Tel 061 225 2249

SOFTWARE

GAMES (Arcade)

Frogger	£24.95	Q*Bert	£24.95	Midnite Mason	£24.95
Micro-Surgeon	£19.95	Fathom	£19.95	Moonsweeper	£19.95
Demon Attack	£19.95	Star Trek	£19.95	Buck Rogers	£19.95
Hopper	£14.95	Munchamn	£ 9.95	Indoor Soccer	£14.95
Miner 2049'er	£23.95 Very few of this superb game left Out of production.				
T.I. Invaders	£ 9.95	Connect 4	£ 9.95	Tombstone City	£ 9.95
Sewermania	£19.95	Superfly	£19.95	Meteor Belt	£19.95
Space Bandits	£19.95	Bigfoot	£19.95	Arcturus	£48.00

SOFTWARE

Adventure games

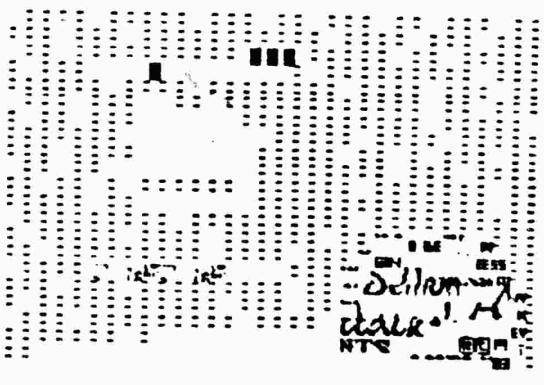
Scott Adams
Return to Pirate Isle £19.95- With graphics, can be printed as you go.
Adventure Pirate £19.95- Module and tape

INFOCOM The ultimate in adventures, almost intelligent. I'm hoping to have them in time for the show, but that's not guaranteed. All have been ordered, but I advised that there may be staggered delivery dates. I'll let you know more when I know myself.

ZORK I	£39.95	ZORK II	£44.95	ZORK III	£44.95
ENCHANTER	£39.95	DEADLINE	£49.95	WITNESS	£39.95
STARCROSS	£49.95	INFIDEL	£49.95	PLANETFALL	£39.95
SORCEPER	£44.95	CUT THROATS	£39.95		

And announcing : HITCH HIKERS GUIDE TO THE GALAXY £39.95 - There has been some queery about Hitch Hikers- I am assured it is in the first batch being sent.

AA



2021- yes, Howard did send in the jumbled up screen print above for his advert.
His paper had a large TI99/4a image as a background which does not show well here.

ARCADE HARDWARE

214 Horton Road Fairweather Manchester M14 7QE Tel 061 225 2248

PERIPHERALS

T.I. 32k R.A.M. Cards £95.00
T.I. INTERNAL DISC DRIVE £95.00
MYARC DISC CONTROL CARD £185.00 - Conrtols up to 4 double sided double density disc drives. Also uses T.I. format so no compatibility problems. I've used this card and am very impressed.
MYARC RS232 Card £125.00 - 2 x RS232, 1 x True Centronics.
MYARC 128k R.A.M. card £249.95 - 32k R.A.M. + 96k RAM DISC/ Print Buffer.
MYARC Mini Peripheral Box - £495.00. 1 x RS232, 1 x True Centronics, 32k R.A.M., + all features of Myarcs disc controller + space and power supply for 2 x half height disc drives. A very impressive and compact unit.
Disc Drives - With the wide range of drives now accessible by the Myarc Controller, it is no longer possible to carry a full range. Please call for prices and advice.
BOXCAR RS232 £119.95 - Gives 1 x Parallel, 1 x RS232 without the need for the peripheral box.
BOXCAR 32K R.A.M. - £125.00 Provides full features of T.I. 32k card, without requiring the T.I. box.
AXIOM Centronics - £109.95 Gives 1 x True Centronics output. Can be used with the top printers. Stand alone.
Alphacom 42 Thermal Printer - £145.00. 40 column printer just plugs into computer.
NAVARONE WIDGIT £39.95 - Plug in three modules and save wear on your cartridge port. This item is usually in short supply, but always on order if none in stock.
Personal Periperals SUPER SKETCH £65.00. - Do freehand and traced drawings on screen, save to tape. See serious software for screen dump and disc save program.
Personal Peripherals Joysticks (Pair) £24.95. Very robust, smooth action.
Newport Controls PROSTICK (single) £22.00 . The best joystick I stock, with adaptor for JOYST 1 OR 2.
SUPER CHAMP JOYSTICK (single) £14.95. Budget price joystick.

SERIOUS SOFTWARE

NAVARONE CONSOLE WRITER £49.95
NAVARONE DATABASE £65.00 - Module + Disc.
HOUSEHOLD BUDGET MANAGEMENT £9.95
HOME FINANCIAL DECISIONS £9.95
T.I. WRITER £74.95 The ultimate word processor.
MULTI-PLAN £74.95 The ultimate spreadsheet.
T.I. LOGO II £74.95 The child appropriate language
DFX SCREEN DUMP/DISC SAVE £24.95 Dump Super Sketch drawings to printer/save them to disc.
DISC REPAIR KIT £19.95 Arcades first program, a replacement for Navarones Disc Fixer. It's cheaper, better documented, more user friendly, easier to use.

EXTENDED BASIC £69.95 Independantly manufactured, with improved manual. Licenced from T.I. and identical in use.

I LOOK FORWARD TO SEEING YOU ALL IN BRIGHTON ON 28TH APRIL

Proprietor Howard Greenberg

VAT Reg No 383 3080 57

BULLETIN BOARD

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

OTIUer DAVE HEWITT of HEADINGTON, OXFORD (not to be confused with another Dave Hewitt, one-time contributor) has the following for sale:

1 TI console	}	
1 Yahtzee module	}	£50.00
1 Music Maker module	}	

Either contact Dave direct (he is a Contact) or contact him through me.

GORDON PITT, of the West Midlands TI User group, has an Expansion Box, a Disk Controller card, and an "internal" Shugart single sided, single density drive for sale at £250.00. Contact him through me.

MICK SAYERS, of the dissolving DERBY TI USER group, is offering a number of items for sale:

Speech Synthesizer	£22
Pole Position	£10
TI Invaders	£ 6
Early Reading	£ 3
Beginning Grammar	£ 3
Addition + Subtraction	£ 3
Demonstration Module	£ 1

Books:

Programming In BASIC	£ 5
Kids & The TI-99/4A	£ 4

Contact him on 0332 676907. These items are all 'or nearest offer'.

CHARLES LACEY is still looking for Speech Synthesizers, Terminal Emulator IIs, and also a set of cassette leads. Contact him through me. (And yes, I will be telling him about the items above!)

L O A D S O F U S E

A LIST OF CALLS COMPILED FROM OTHER NEWSLETTERS

SASKATOON TI COMPUTER CLUB (STICC)

From TERRY ATKINSON of NOVA SCOTIA, requiring Extended BASIC and the 32K:

```
10  CALL CLEAR :: CALL INIT
20  CALL LOAD(8196,63,248)
30  CALL LOAD(16376,67,85,82,83,79,82,48,8)
40  CALL LOAD(12288,255,129,129,129,129,129,129,255)
50  CALL LOAD(12296,2,0,3,240,2,1,48,0,2,2,0,8,4,32,32,36,4,91)
60  CALL LINK("CURSOR") :: END
```

After running this program, until QUIT or BYE occurs the cursor should remain redefined. To provide other shapes change the values in line 40. The latter 7 numbers define the shape in decimal. Create your shape in binary/hex and then convert to decimal.

The WINNIPEG USER GROUP apparently suggests using 0,0,0,0,0,0,0,252 for a flat line (or perhaps 0,0,0,0,0,0,0,255 ?). You could be really mean and use either 8 zeros or just delete line 40 altogether. The COLUMBUS OHIO USER GROUP suggest 48,48,63,255,254,124,24,12.

The STICC newsletter goes on to give a whole list of CALLs to be used with the 32K and Extended BASIC:

CALL LOAD(-31806,N) where if:

```
N =   0  :  restores to "normal mode"
N =  16  :  disables FCTN=
N =  32  :  disables sound
N =  48  :  disables FCTN= and sound
N =  64  :  disables sprite motion
N =  80  :  disables sprites and FCTN=
N =  96  :  disables sprites and sound
N = 128  :  disables all three
```

CALL LOAD(-31748,N) where if:

```
N =   0  :  cursor blink & sound are OFF
N =   1  :  cursor blink & sound NORMAL
N = 100  :  very fast blink and sound
```

```
CALL LOAD(-31804,0,36)          similar to BYE
CALL LOAD(-31878,N)             disables N sprites
CALL LOAD(-31888,63,255)        disables disk drives
CALL LOAD(-31888,55,215)        restores drives when followed by RUN, NEW
                                or editing
CALL LOAD(-31962,255)           reboots LOAD program from disk
CALL LOAD(-31868,0,0)           disables 32K
CALL LOAD(-31868,255,231)       restores 32K
```

*2021- Note that due to partial decoding of the memory addresses, the following addresses actually refer to the SAME memory location:
-31931, -32187, -32443, -32699
These are each 256 bytes different.
This applies to all similar addressing.

CALL LOAD(-31931,0)

disables Extended BASIC's protection (I had some qualms about whether this LOAD ought to be restricted, but that would have conflicted with my principle that 99ers have a right to any and all information about their computer.

In fact, the important "bit" is the Most Significant Bit (i.e. leftmost). Therefore, any value less than 128 will disable protection, while any value of 128 and over will institute it. PB)

CALL LOAD(-32699,0)

This too is said to disable protection

Others encountered include:

CALL LOAD(-32116,4)

Transfers the system from Extended BASIC to TI BASIC. I understand that the 32K should not be present for this to work.

2021- without 32k ram you cannot use CALL LOAD.
This CALL must be in COMMAND mode, not in a program.

P E E K S A N D P O K E S

A LIST OF CALLS COMPILED FROM OTHER NEWSLETTERS

SASKATOON TI COMPUTER CLUB (STICC)

CALL PEEK(-28672,A) where if:

A = 96 : Speech Synthesiser is present
A = 0 : Not present

CALL PEEK(-31808,A,B)

provides pseudorandom integers in the range 0,254

CALL PEEK(-31878,A)

the value in A represents the highest numbered sprite

CALL PEEK(-31952,A) where if:

2021: A= 255 not 55

A = 55 : the 32K is ON
A = ? any other value means it is OFF

CALL POKEV(-32768,0) "normal" mode
CALL POKEV(-32353,0) screen is "transparent"
CALL POKEV(-32272,0,"",-30945,0) 40 column screen
CALL POKEV(-32280,0) multiple colours
CALL POKEV(-32766,0) "bit-map" mode

THE TI 99/4A PERIPHERAL EXPANSION BOX DISASSEMBLY PROCEDURE

by John B. Colson who assumes no responsibility if anyone gets in trouble with this information.

Do you need to get inside your PEB? Some have wanted to increase the power output of the power supply. One reason for more power is to allow using half height disc drives that require more than the 12 volt 500 millia.peres available from the TI supply. Maybe you have some foreign material stuck inside. Maybe you need to do maintenance on this unit. Whatever the reason, to many observers the disassembly appears mysterious. The clue is that the sides and front are removed as a single unit.

If you are going to upgrade the power supply output, you need to have obtained in advance the necessary components. These components depend upon how much power you are going to need. The voltage regulator that is in the TI unit can give one ampere if properly heat sunk. For one and one half amperes, a Motorola MC7812C can be used. For three amperes, a Fairchild ua79H12KC can be used. For five amperes, a $20.00 RCA SK9341/933 should work. To provide a good heat sink I used the fan side of the power supply housing.

To start this procedure: shut off the power and allow the capacitors at least a full minute to discharge internally instead of into you. Remove the power cord. Remove the disk drive(s). Take off the top cover by pushing the two back clips forward and tilting cover forward. If you have external cables attached to peripheral cards they are to be disconnected, i.e. printer, external disk drive, etc. Disconnect the internal disk drive ribbon cable and remove all cards from the box. The power supply is on the left side of the box as viewed from the front. This is obvious, but as we turn the box over, we need to keep track of where the power supply is located. Remove six screws from the bottom of outside flanges and one screw from the bottom of the power supply housing. Remove one screw from each end of the box. Place the box on its face on a protective, soft, (non-scratching) surface and remove three screws from the back of disk housing and three screws from back of power supply housing. Now the two pieces are separated and you should be able to lift the back and bottom with the heavy power supply straight up and away from the face and sides.

All the above voltage regulators come in TO3 packages and will mount at 45 degrees in the lower outside corner of the fan side. To perform this drilling, the power supply circuit board should be removed. This is accomplished by loosening two phillips screws at the inside base of the plastic right angle bracket holding the circuit board and sliding the bracket with the board to the outside. By snapping the ears on each of three plastic connectors away from each other, the connectors can be tilted toward the foil side of the board and removed. Now the remaining cables can be put out of the way under the fan and the drilling can proceed. To play safe from case ground, I electrically insulated the voltage regulator and used silicon pad and heat sink compound on my mount. Carefully! remove the voltage regulator (the only TO3 package) from the TI circuit board and attach three wires that will reach your newly mounted voltage regulator. Note well where each wire belongs and attach them to externally mounted voltage regulator (with pins inside). Now you may re-assemble and see if it works. Mine did.

GETTING THE MOST OUT OF TI-WRITER

by Allen Burt.

This time I would like to start with page 98 of the TI-WRITER Manual -"SPECIAL CHARACTER MODE" . There are two potentially misleading statements on this page.

1) The third paragraph implies that you have to use the TRANSLITERATE command if you wish to use a combination of two or more special codes. This we now know to be untrue. In the last article I demonstrated how to input three character combinations.
2) The second error is the page reference at the end of paragraph three(...see example on page 121....should read 107).

REMINDER

SPECIAL CODES are accessed by means of CONTROL key and KEY 'U' -the cursor then appears as CHR(95) the underline character. You can then enter the special code by pressing either SHIFT or FUNCTION together with the KEY as shown on page 146 of manual.(This is only necessary to obtain ASCII codes 0 - 31).

MORE TIPS FOR THE "EDITOR".

a) Space can be saved on TI-WRITER files by removing all the blank lines within the layout. Once you have decided how many blank lines are needed for each part of the document you can place a number of LINE FEEDS at the end of the line preceeding the blank line area with the aid of CONTROL 'U' & SHIFT 'J' -ie CHR(10),(this document is produced without any blank lines on the screen display). This means that one record in the file can be used for one line of characters together with a number of blank lines.
   Each line of writing on the screen display requires one record in the file (Files are Display Variable 80 which means that each record can contain up to 80 Characters -each blank line occupies one record).

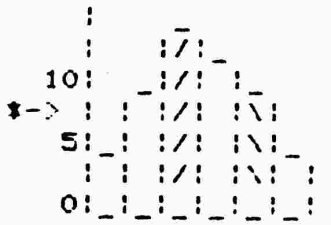
b) Sometimes when finishing a line there is not enough space to enter the Carriage Return(CR) without word-wrap carrying the last word and the CR over to the next line. This can be avoided by using CONTROL 'U' & Shift 'M' [CHR(13)] -but you will have to ensure that you are not on the line above...."*End of file Version").... because this method of inputting a CR will not put you on the next line. You have to 'Insert' a line first and then use the 'arrow keys' to move between lines.
   This method of using the CR is particularly useful when editing data tables. It does not produce an extra line that has to be deleted afterwards.

c) Most printers can be prevented from stopping when the 'paper out' signal is activated. The Gemini 10 requires CHR(27);"8" codes. This is useful when printing on single sheets because it allows you to print to the bottom of the page. Make sure that you do not have too many lines otherwise you will print on the platten. This is especially important if you are using tip (a) -the line numbers at the side of the screen will not correspond with the actual number the printer will print.

EXTEND THE USE OF TI/WRITER

TI-WRITER can be used for much more than just producing letters ie - a substitute for a typewwriter. In the last article I described how to make use of the CONTROL 'U' function in the Text editor mode. This function can be used to extend the application of the system and to produce integrated documents of words and diagrams.

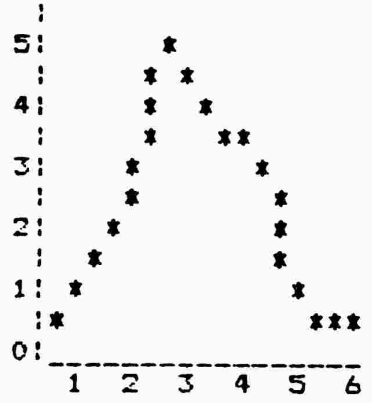
For example it is easy to show a Histogram (Bar chart) like this:



This just uses the CHR(124) obtained by using 'FUNCTION'[F] and KEY 'A' for the verticals and the underline character CHR(95) (-[F] and KEY 'U').

A useful tip when doing this type of exercise is that if you place the CHR(124)'s in the appropriate locations and wish to continue them downwards from the point indicated by the asterisk -just move the cursor down to the next line and press CONTROL [C] and KEY '5' this copies the line above on to that line. When you draw diagrams like this, it is better to insert a number of lines in order to have room to manoeuvre.

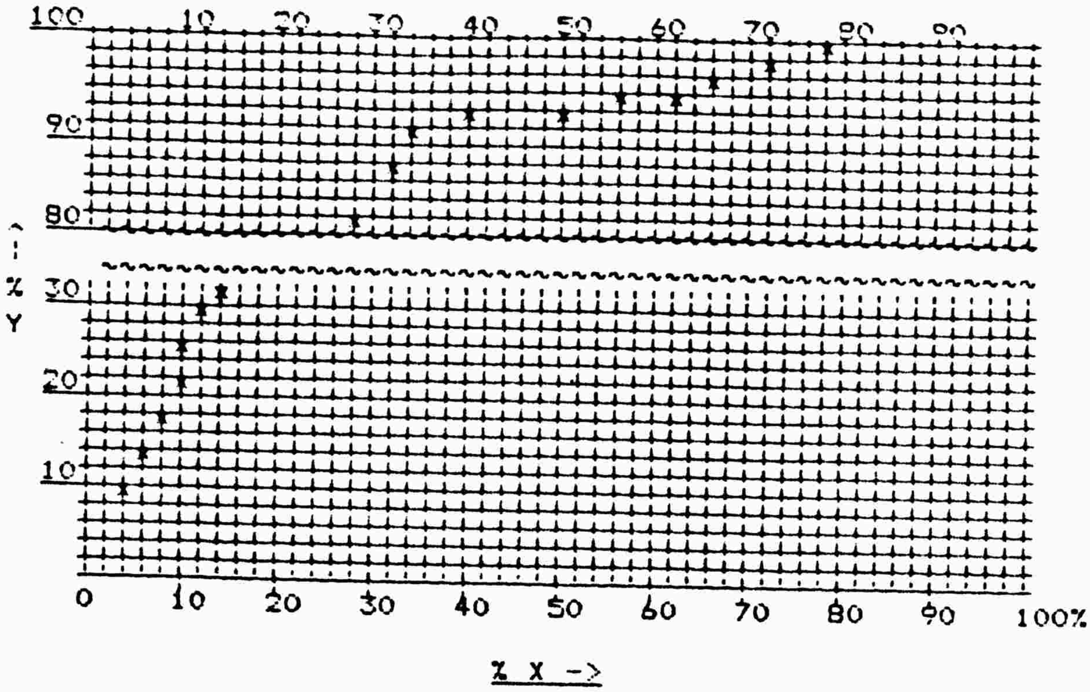
If you want to include a simple graph within your script try doing this:



A more sophisticated graph can be achieved using the above techniques. In the next example I found the 'COPY' command very useful because having once obtained the required width - I only had to 'copy' down the required number of lines using [Control & KEY '5'].

REMEMBER that when you place a set of special codes at the start of the line the space they occupy will not recognised by the printer. That is the printed line will commence at the location of the first special code. This can place the numbers used in the graph in the wrong place. You have to enter your special codes at the point you wish the following characters to print. Thus what you see on the screen is not necessarily what you will get on the printout.

USING TI-WRITER TO DRAW A GRAPH



TI-WRITER used to draw graphs. The horizontal lines are achieved by setting printer into underline mode[CHR(27);CHR(45);CHR(1)]. The line spacing is set to 7/72" [CHR(27);"A";CHR(7)] -this approximates to 1/10". If a CARRIAGE RETURN is placed at the point where the line should finish, the printer will draw a line to that point.

The vertical lines are drawn by using CHR(124) -Function 'A'. As the printer normally prints at 10 CHRS to the inch this will produce a grid of roughly 1/10" squares.

There are two points to watch :

a) If you do not want the underlining to start at the beginning of the printer line the underline code must be placed at the start of each line and cancelled at the end of the line before the Carriage Return. There is another means of achieving this and that is to set the left hand margin to the required position(on GEMINI this is CHR(27);"M";CHR(n) -n being the column to start printing). THIS CAN ONLY BE DONE BY USING THE PRINTER CODES NOT BY SETTING TI-WRITER's TABS.

b) The second point is that many printers do not align the characters in bi-directional mode. You are advised in the Manual for TI-WRITER that for tabulation it is advisable to set the printer to Uni-directional printing.

(How a line will appear on the screen display.)

'b-1 90!!!'b-0cr

↑ (actual start point for printing)

C A L L L O A D S

CALL LOADS is a regular column in SUBFILE99 featuring various CALL LOADS and CALL PEEKS that can be used on the TI Home Computer. In order to use these LOADS and PEEKS you must have the ED/ASSM, X-BASIC or MINI-MEMORY Modules and 32K Memory Expansion. Do you have interesting LOADS or PEEKS that you have found? Why not send them in to SUBFILE99 (TI5361) and share them with your fellow TI'ers?

GETTING A LINE ON YOUR PROGRAM

Have you ever wondered how your TI Console stores and reads the BASIC program you type in? Well, I have! It all started when I began to have problems with one of my consoles. Every time I EDITed an existing line, other lines got "messed up" or even MOVED to some other part of the program! Eventually, I had to have the console replaced, but it didn't keep me from becoming curious. I was determined to figure out how this could happen. When I finally found the answer, I learned a great deal about how the TI keeps track of the program we type into it. Below is some info that you can use to get "computer's-eye-view of the program you have typed in.

IT TAKES TWO TO TANGO

Any BASIC operating system has to keep track of every line of code you type in. It not only has to keep track of the program data, but it also has to keep track of the line numbers, too! The TI BASIC system uses two different areas for this: 1) the Line Number Table and 2) the Program Area.

All this info is stored in the highest memory available to the console. For TI BASIC that will usually be somewhere around address 14228 and for X-BASIC with MEMOR EXPANSION, somewhere around address -28. This all depends on the number of files open (as in CALL FILES(1)) and whether the disk drives are attached, etc.

PROGRAM FIRST

The actual lines of program code are stored first. Each line of code is stored in a series of bytes. Most all reserved words like PRINT, INPUT, REM, etc. are stored as a single byte value called a TOKEN. For instance, the TOKEN values for the above three words are 156(>9C), 146(>92) and 154(>9A). TI BASIC also keeps track of variables, constants and jump references in a similar way. Each line is preceded by a length byte and, at the end of each line, the value 0 (zero) is placed to indicate the end of the line.

As each line is typed in, the BASIC operating system translates it into these byte values and stores it at the top end of the available memory.

When you EDIT a line or INSERT a new line between existing ones, that new line is stored at the "end" of the Program Area, not inserted into the middle. This is important to remember when trying to locate a line of code! Note also that the line numbers (i.e. 100, 110, 120, etc.) are NOT stored with the line of code. This allows the TI system to easily RESequence your program, and allows the system to place lines at any available memory location, instead of in execution order only. So where are the line numbers? In the Line Number Table, of course!

THE LINE NUMBER TABLE

After all the lines of code are stored, the Line Number Table begins. Each line of code has 4(four) bytes of information in the Line Number Table; two bytes for the line number (>0064=line 100), and two bytes for the memory address where the line of code is stored (>FFE7= 28). When the BASIC interpreter is running the program it looks at the Line Number Table, gets the line number and the line address, branches to the address, gets the actual program line and THEN performs the line of code! Not so simple, eh?

Since the Line Number Table is stored at the end of all the program lines, each time you add a new program line, the entire table has to be "shifted" downward into lower memory. This is why, especially in a long TI BASIC program, it takes quite a while for the cursor to "come back" after you have EDITed a line. The operating system must code the line, shift all the line numbers and addresses downward and then insert the new line number and address into the table. That would take anybody a little time!

PEEKING AROUND

Let's look in on the operating system as it does all this stuff! Here's what we'll do:

1) Enter X-BASIC (or TI BASIC with ED/AS or MINIMEM module) and type:

```
100 PRINT "THIS IS A TEST"
```

2) Find Line Number Table:

To find the Line Number Table, PEEK into the CPU RAM PAD at address -31950; get two consecutive bytes; convert these bytes to decimal; subtract 3 bytes (overhead) and, if you are using X-BASIC with MEMORY EXPANSION, subtract 65536: you now know the address where the line table starts!

```
CALL PEEK(-31950,A1,A2)
LT=A1*256+A2-3-65536
```

In X-BASIC, the value should be -47 and in TI BASIC it should be 14273. As each line of code is typed in, the table will move farther down into memory.

3) Get the line and address:

Using the start of the Line Number Table, get the first line of code and the address where it is stored. In X-BASIC this is stored in MEMORY EXPANSION so you use the following:

```
CALL PEEK(LT,L1,L2,A1,A2)
LN=L1*256+L2
LA=A1*256+A2-65536
LT=Line Table Address ( -47)
LN=Line Number        ( 100)
LA=Line Address       ( -42)
```

In TI BASIC, the program is stored in the console itself, therefore you must look into the VDP area with the command PEEKV:

```
CALL PEEKV(LT,L1,L2,A1,A2)
LN=L1*256+L2
LA=A1*256+A2
LT=Line Table Address (14273)
LN=Line Number        (  100)
LA=Line Address       (14278)
```

4) Get the line length:

Each line of program code is PRECEDED by a length byte. This tells the system how many bytes to interpret. To find the line length, read the byte PRECEDING the Line Address:

```
X-BASIC: CALL PEEK(LA-1,LL)
TI-BASIC : CALL PEEKV(LA-1,LL)
In both cases, LL=18.
```

5) Read the Program Line:

Now all we have to do is read in 18 bytes of code starting at the Line Address (LA):

```
X-BASIC:
CALL PEEK(LA,A,B,C,D,E,F,G,H, I,J,K,L,M ,N,O,P,
PRINT A;B;C;D;E;F;G;H;I;J;K; L;M;N;O;P;Q;R
TI-BASIC:
CALL PEEKV(LA,A,B,C,D,E,F,G,H, I,J,K,L,M,N,O,P,
PRINT A;B;C;D;E;F;G;H;I;J;K; L,M;N;O;P;Q;R
```

In both cases, the values A-R will equal: 156 199 14 84 72 73 83 32 73 83 32 65 32 84 69 83 84 0

```
156 - Token value for PRINT
199 - indicates a string value
 14 - length of string
```

The rest of the bytes are ASCII values for the string (84=T, 72=H, 73=I, 83=S, etc.).

POKING AROUND

Now let's see if we can't reverse the process. This time we'll do a few POKES into memory that will make the computer think you typed in a line of code! Here's what we'll do:

1) Clear memory and type CALL INIT
2) Set up Line Table:

The system needs to know where the line table starts, so:

```
CALL LOAD(-31950,A1,A2)
X-BASIC: A1=255, A2=220
TI-BASIC A1= 55, A2=204
```

3) Insert line no. and address:

Next we need to fill up the line table with line and address data:

```
X-BASIC:
CALL LOAD(-39,0,100,255,222)
TI-BASIC:
CALL POKEV(14281,0,100,55,206)
```

4) Insert line length and line code:

Now all we need to do is insert the line data:

```
X-BASIC
CALL LOAD(-35,10,156,199,6, 72,69,76,76,79,33,0)
TI-BASIC:
CALL POKEV(14283,10,156,199,6, 72,69,76,76,79,33,0)
```

5) LIST YOUR PROGRAM!

Because the system has not been properly informed as to where the program ENDS, this is not "RUNable" code. But at least you can see your work!

EXAMPLE PROGRAM

Below is a program that will "read itself". Using the tools we learned above, this routine will read the Line Number Table, and find the Line Address and Line length. After each line is found, the program waits for you to hit a key before it reads the next line.

NOTE: This program is written for TI BASIC with ED/A or MINIMEM modules. To use it with the X-Basic and MEMORY EXPANSION you need to make a few changes:

1) Line 220 - add -65536 to the end of the line
2) Line 260 - change PEEKV to PEEK
3) Line 290 - add -65536 to the end of the line
4) Line 330 - change PEEKV to PEEK

2021 CORRECTION- above 4 lines should refer to LINE 250, 290, 320, and 360

Armed with this info, you can begin to learn more about how the TI system works and you can use these tools to help read AND write your own programs. A number of programmers place routines like this into their programs as a way of "protecting" valuable code sequences or algorithms from prying eyes. It's usually called "embedded code". You can also use routines like this to translate a text line into program format once you learn to decode all the token values!

```
120 REM
130 REM  PROGRAM PEEKER
140 REM
150 REM
160 REM  11/84
170 REM  SUBFILE99
180 REM
190 CALL INIT
200 CALL CLEAR
210 REM
220 REM  FIND LINE TABLE
230 REM
240 CALL PEEK(-31950,A1,A2)
250 LT=A1*256+A2-3
260 REM
270 REM  GET LINE AND ADR
280 REM
290 CALL PEEKV(LT,L1,L2,A1,A2)
300 LN=L1*256+L2
310 IF LN=0 THEN 570
320 LA=A1*256+A2
330 REM
340 REM  GET LINE LENGTH
350 REM
360 CALL PEEKV(LA-1,LL)
370 REM
380 REM
390 REM  PRINT THE DATA
400 REM
410 REM
420 PRINT "LINE";LN
430 PRINT "ADDR";L
440 PRINT "LEN";LL
450 PRINT
460 CALL SOUND(150,1400,0)
470 CALL KEY(0,K,S)
480 IF S=0 THEN 470
490 REM
500 REM  UPDATE LT
510 REM
520 LT=LT-4
530 GOTO 290
540 REM
550 REM  END
560 REM
570 PRINT "END OF PROGRAM"
580 PRINT
590 END
```

'POWER TO THE PEOPLE.....'
[and all that sort of thing]

Tony Ralph

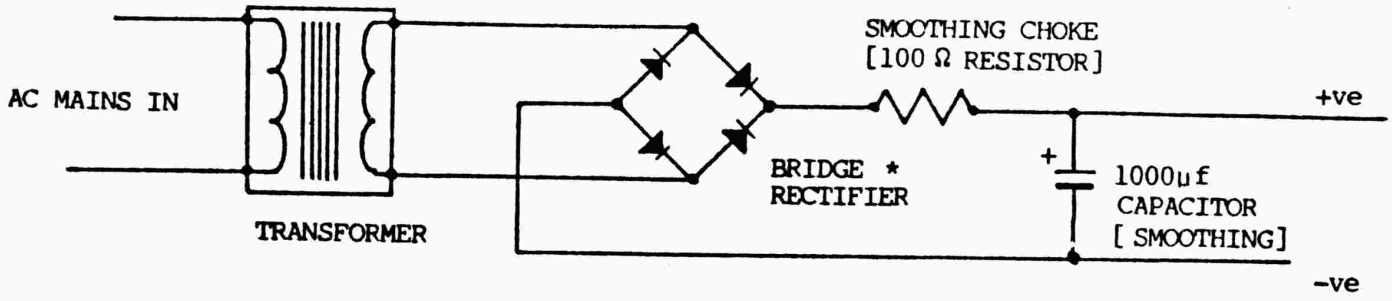
Whilst talking to Pete on the telephone last week about disk drives and related topics, we chanced upon the subject of power supplies. To those of you familiar with Brooko's style it will come as no suprise that I was quickly talked into writing a few lines for TI-LINES about what I know of power supplies. Suffice to say that as I know next to nothing about designing power supplies, these notes are going to be extemely short!

The first point to bear in mind is that what I shall decribe is a non-stabilised type of DC supply. The user is at the mercy of the CEGB and what they squirt down the electric string; the rising and falling of the AC Mains supply directly affecting the DC output of the power supply circuit.

Incidentally if you do not understand any of the terms used in this article please write and say...it may encourage further notes [it may not of course].

There are three elements to the basic circuit. Firstly, a means of chopping down the 240V AC Mains supply to something like wot we want. This is done with a transformer. There are transformers made for just about every conceivable requirement so you should be able to get just what you want [famous last words]. Secondly we need to convert this transformed AC supply into DC [direct current as opposed to alternating current]. Enter the famous Bridge Rectifier. These days such items come as blobs of plastic with four wires sticking out, but my little diagram shows the old fashioned way of producing the rectifier; connecting the positive ends of two simple diodes together and then connecting the negative ends of those diodes onto the positive ends of two more diodes and finally connecting the second two diodes' negative ends together [phew!]. If in doubt look at the diagram [or turn the page to the next item altogether!]. Finally, although we now have a DC supply avaiable, we will want to smooth it out as it is very spiky and noisy. To do this we stick a resistor in the positive line, call it a choke [no, I don't know why] and then connect a fairly large capacitor across the positive and negative lines, being sure to connect the positive side of the capacitor to the positive line [Often the positive side of the capacitor is marked with a red blob of paint...but not always].

Voila' as they say in France; a DC power supply. This sort of thing is useful for running a car radio off the mains or a simple audio amplifier or even a CB set [which I did]. However as I mentioned at the beginning, it is a non-stabilised supply and would not, as it stands, be suitable for things like stereo radio tuners and I am not sure it would be suitable for disk drive power supplies either. Perhaps somebody can take this on a bit and tell us all how to organise a stabilised supply.



* four diodes connected together as a bridge.

F O R T H T I P

Reprinted from the Central Iowa Users Group Newsletter

For those of you who have a one disk drive system .... this program is for you! This will be put in our library soon!

It will copy the entire contents of an original disk in only 3 passes! For example, if you had a disk with 20 program or data files on it and used TI's DISK MANAGER, (and only one disk drive) it would take you about 8 minutes to copy all of the files. With two drives, it takes about 3 minutes. Now you can do it in 2 minutes! First, type and save this program.

To operate, load TI FORTH, and when the cursor appears, type COLD. The disk drive will "kick in" momentarily. Next, insert your disk with this program on it (drive No. 1), and type 30 LOAD. This will load both screens automatically. The screen prompts will give you all of the instructions to proceed.

```
SCR  No. 30
  0  ( 3 PASS DISK COPIER - DOUG SMITH 301-645-1432 )       : IT ;
  1  : CLS 16 SYSTEM ;  : VMBW 2 SYSTEM ;  : VMBR 6 SYSTEM ;
  2   0 VARIABLE AREA 15360 ALLOT 0 VARIABLE PL 0 DISK_LO !
  3  : TX CLS 5 11 GOTOXY ;
  4  : M1 TX ." INSERT COPY DISK-PRESS ANY KEY " KEY DROP ;
  5  : M2 TX ." INSERT  MASTER - PRESS ANY KEY " KEY DROP ;
  6  : M3 TX ." DONE - ENTER W TO COPY ANOTHER " ;
  7  : PR TX ." COPIER NOW READY-PRESS ANY KEY " KEY DROP ;
  8  : PU PL @ 20 + PL @ 5 + DO I BLOCK AREA 2 + I PL @ 5 + - 1024
  9       * + 1024 CMOVE LOOP ;
 10  : BU PL @ 5 + PL @ DO I BLOCK UPDATE LOOP M1 FLUSH ;
 11  : DR PL @ 10 + PL @ 5 + DO AREA 2 + I PL @ 5 + - 1024 * + I
 12       BUFFER 1024 CMOVE UPDATE LOOP FLUSH PL @ 15 + PL @ 10 + DO
 13       AREA 2 + I PL @ 5 + - 1024 * + I BUFFER 1024 CMOVE UPDATE
 14       LOOP FLUSH PL @ 20 + PL @ 15 + DO AREA 2 + I PL @ 5 + -
 15       1024 * + I BUFFER 1024 CMOVE UPDATE LOOP FLUSH ;    -->

SCR  No. 31
  0 ( 3 PASS COPIER CONTINUED )
  1 : BPU PL @ 28 + PL @ 20 + DO I BLOCK 5120 I PL @ - 1024
  2       * + 1024 VMBW LOOP
  3       PL @ 28 + BLOCK 3072 1024 VMBW
  4       PL @ 29 + BLOCK 1122 1024 VMBW ;
  5 : BDR PL @ 25 + PL @ 20 + DO 5120 I PL @ - 20 - 1024 * + I
  6       BUFFER 1024 VMBR  UPDATE LOOP FLUSH
  7       PL @ 28 + PL @ 25 + DO 5120 I PL @ - 20 - 1024 * + I
  8       BUFFER 1024 VMBR  UPDATE LOOP
  9       3072 PL @ 28 + BUFFER 1024 VMBR UPDATE
 10       1122 PL @ 29 + BUFFER 1024 VMBR UPDATE FLUSH ;
 11 : PAS BPU PU BU BDR DR ;
 12 : CY  0 PL ! 30 / 1+ 0 DO PAS PL @ 50   IF M3 ELSE M2 I 2+ .
 13       30 PL +! THEN LOOP ;
 14 : W   M2 89 CY ;
 15 PR W
```


2021 Note: Please see Page 59 for correction and note

CLOSE FILE

For a number of reasons I was unable to complete my task of compiling this issue of TI-LINES before the Easter break, when I had arranged to go away, and this has resulted in yet another delay in production. (It is now 2 a.m. on the 16th to give you some idea of the time-scale). A number of events have all managed to carefully combine to produce a series of near-calamities, and the upshot is that I will shortly be moving house - again - and going through the usual agonies of postal delays, misdirections, and the inevitable total collapse of my carefully-cultivated filing system (I remember where I chucked something, and as long as no-one tidies up, I know where I am).

I apologise profusely in advance (and probably in arrears) for the inevitable inconvenience that this will cause to everyone, and hope to be back on my usual uneven footing by June at the earliest (there's honesty for you!).

Because of these changes my attempts to find a solution to the TIHOME Software Collection dilemma have come to naught, and I am being indirectly (and, it must be said, unintentionally) pressured to come up with a solution fast.

Therefore, in the interests of expediency (until a better solution presents itself) I intend to attach the TSC to OXON TI USERS as soon as is practical.

If anyone has any objections or observations on this move (bearing in mind that OTIU, like TIHOME and TIHCUC before it, is not a democratically-elected club, but is a commercial concern owned and run by me for the benefit of TI Users generally - and not, hopefully, to their detriment or disadvantage), I would be pleased to receive them. Access to TSC would be through membership of OTIU, either as an Oxon resident or an Associate subscriber. Existing OTIUers would therefore be entitled to obtain TSC programs should they so desire. I cannot emphasise too strongly the fact that TSC will not knowingly contain any commercially-produced software of any description.

If I cannot arrive at a satisfactory solution, then TSC will cease to exist by the end of 1985.

It seems that I made an error in a previous TI-LINES when I said that membership of OTIU ought to be sufficient to gain you free admittance to the National TI User Show in Brighton (as it did in Manchester last year). I have arranged with Clive Scally that your entry will still be free, and I intend to honour my undertaking to pay the entrance fee for those individual OTIUers attending on Sunday. I look forward to seeing as many of you as can make the trip.

I must apologise for the scrappy layout of this issue; I decided to put out as much information as I could and leave the explanations to another time. I hope that this does not upset too many readers - the best thing to do is to put it to one side against the time that you do come to grips with it all (with or without my assistance!).

Good programming,

Peter Brooks

This correction and note regarding the Forth article on Page 57 was published in TI-LINES v1 n14 May 1985:

If you had trouble with the Forth screens on page 57, well, part of the problem was a ")" symbol missing from line 12 of screen 31. It goes between the PL @ 50 and the IF M3 ELSE. However, when I correctly entered both screens and tried to use them as suggested, they did not function correctly. I got as far as the instruction to INSERT MASTER - PRESS ANY KEY whereupon the drive loaded - what I don't know - and then locked up. Any Forth mechanics out there ?

This page is blank