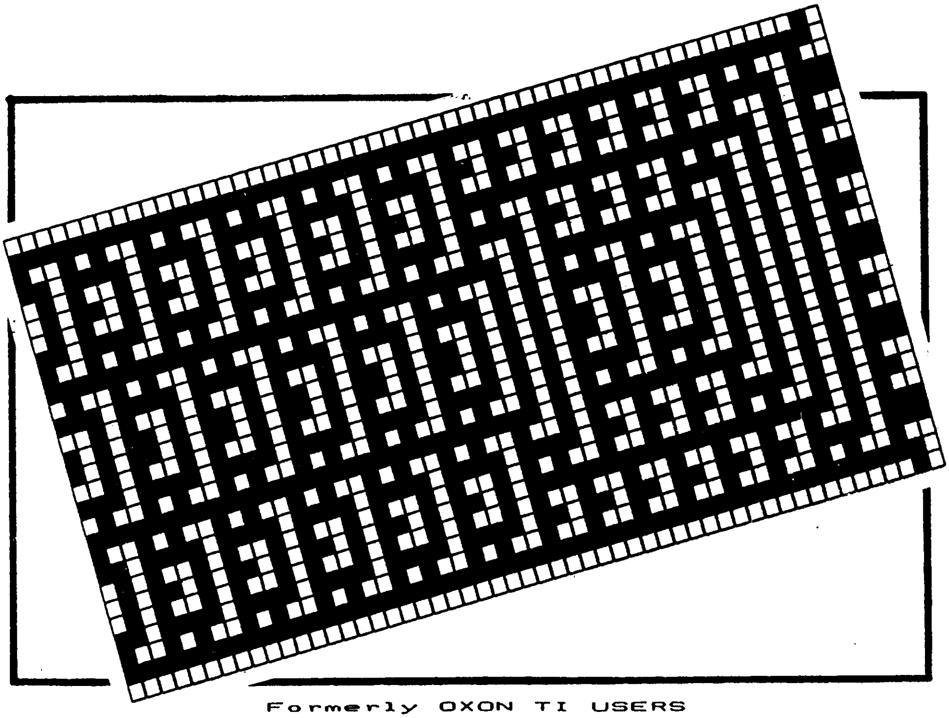
INTERNATIONAL TI-LINES

Volume 4, Issue 5 October 1st., 1987



Formerly OXON TI USERS

A PUBLICATION OF THE INTERNATIONAL TI USER GROUP

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

PETER G. Q. BROOKS
96 BANBURY ROAD
OXFORD OX2 6JT

OXFORD 510822

Index to articles

PLEASE NOTE: AS THE STOCKS OF ROMOX CARTRIDGES ARE NOW LOW, FROM THE 1ST OF NOVEMBER ALL FUTURE ROMOX CARTRIDGES WILL COST £5.00 EACH, BUT WILL COME WITH A PROGRAM ALREADY ONBOARD (CHOSEN BY THE BUYER FROM A LIST TO BE PUBLISHED SHORTLY). THE MAIN REASON FOR THE RISE IN PRICE FROM 50P TO £5.00 IS THE FACT THAT WE HAVE NO RESERVES ON WHICH TO FALL BACK FOR REPAIR, MAINTENANCE, AND PRODUCTION OF NEW CARTRIDGES.

R E A D   T H I S   F I R S T !

IT editorials are tending to become a series of apologetic explanations as to why the planned effects of the previous issue never quite made it to the reader.

This issue is no exception to the growing rule: last issue suffered (as always) from gremlins. The first tribe of gremlins applied their skills to the "new" page outline; I ended up using the wrong outline, which in turn was greeted with glee by the second tribe of gremlins who inhabit the photocopier. These latter pounced on the outline, distorted and smeared it wherever possible, and generally made a pig's ear of it.

Not content with this, Tribe Two attacked the photocopier's fuser unit, causing it to overheat and glue successive pages of photographs to one another (I had to peel apart 250 pages at one stage, losing patches of toner from 1000 photographs), making another pig's ear out of all the beautifully (and expensively) scanned images from Bloxwich. You will have to take my word for it that the original scans were a sight to behold.

But not, it seems, in TI-LINES...

I discovered a small error in my figures in the GRAPHICS REPORT article which appeared in V4.3 On an 8 bit graphic interlace the line feed after the third line should be 22/216 and not 24/216. 24/216 is the total height of each interlaced group, where the print head is, by the third line, positioned at the 3/216 mark from the start. It might make some difference under certain circumstances, so I thought I'd better mention it...

Just to keep you up to date, we are investigating the possibility of adding extra processor cards to the expanded TI system. One of our new subscribers is considering an 8086/8088 board (perhaps enabling us to run IBM-compatible software) and there is a possibility of further cards to take Z80 variants, and also 6502. It would be possible to take the RISC processor from the latest Acorn offering and incorporate it within our system. This isn't something which is likely to be available within the next several months (or even years ?) but it shows that we still have a highly active core working to keep the machine in the running.

In the HANGMAN listing given last issue, there were a number of minor programming slips which I should have picked up earlier and removed. They aren't earth-shattering and don't affect the running of the program but they might confuse relative newcomers to BASIC looking for examples to emulate.

On page 7, line 280: two DIMs aren't necessary; it can all be done on

›››MORE›

one line.

Various places: the use of ASC(CHR$()). For example, where KEY holds a value, and ASC(CHR$(KEY)) has been used to extract the value. As ASC() is the "opposite" of CHR$(), I should have spotted it straight away, but I didn't. Delete the ASC(CHR$()) part and use the variable on its own. If KEY holds 65, then CHR$(65) will give "A", and ASC(CHR$(65)) will give 65. There may be others; eagle-eyed readers are invited to submit any improvements they uncover.

Once again I am suffering postal difficulties. One firm in Milton Keynes has tried on three occasions to send me two disk boxes; finally, after three months, the first pair have just arrived...

On top of this, parcel post charges have gone up by 10p across the board without warning - and without even the paperwork being published so that I can buy a copy of the new price structure. I could have sworn that after the last price increase the GPO pledged not to increase prices again until 1988.

A little dickie bird tells me that paper prices have been falling steadily over the last year, yet I have seen no reflection of this in retail pricing. In fact, MENZIES recently put their prices up on some stationery by a wacking 34%. It shouldn't affect costing of TI-LINES though, since I try to buy materials (like envelopes) in bulk from wholesale distributors, but even so the price difference is not really that great.

DAVID BAINES tells me that his MiniMemory module battery is still functioning FOUR years after installation. Is this a record ?

IAN JAMES recently obtained a copy of a book on Forth which he thought might be of help to him, only to discover - even with his elementary knowledge - that the work seemed to have errors on virtually every page. (For example, the book says that the semicolon is used to print a value from the too of the stack - it should describe the use of the period for this purpose). For the wary, the book is THE BEGINNERS' GUIDE TO FORTH by DAVID JOHNS, published by INTERFACE PUBLICATIONS at £2.95.

Some enquiries placed with me by Ian have prompted me to produce a brief note about disk drives and the capability of the TI disk controller card (as distinct from the standalone disk controller) which should appear elsewhere in this issue.

JOHN SEAGER has asked me if anyone knows how to get TI-Writer to find the ACTUAL page centre when centering text with the .CE command.

I don't, but I know a woman who might...

>>>END

C E L L U L A R   A U T O M A T A

A look at some of the more esoteric applications of computers, as told to intrepid investigative journalist BALDIE BROOKS by Skoorb Retep, representing the Stoidi from the planet !Kreb.

It has been proposed that cellular automata exist naturally in the real world, and searches are under way at several academic sites in various parts of the world. The game of "Life" is a two dimensional example of a cellular automaton, but the simplest - and yet in some ways more complex - form of automaton is the one dimensional cellular automaton (1DCA).

It can be modelled in a number of ways, and one convenient method is to use a computer, which is why the subject is receiving an airing here.

We'll examine the principle of operation, and show how you might implement a 1DCA in TI BASIC. Speed buffs might prefer to encode it in C99, Forth, Pilot, Pascal, or straight 9900 (or even GPL if you've a mind to).

We'll go on from that to examine Conway's Life (2DCA) and its implementation (if you have the VIDEOGRAPHICS module there is a good, if somewhat slow, implementation within that), and demonstrate an Extended BASIC implementation which charts the progress of a single "glider".

It can be made more complex (even by translating into TI BASIC for example!) if the reader wishes to expend some time and effort.

We'll also examine the latest enhancement, which is a three dimensional version of Life (3DCA).

One Dimensional Cellular Automaton (1DCA)

A cube is a 3D model, a sheet is a 2D model, and a line is a 1D model - in this case, a line of cells. The fact that the cells are probably 2 or 3 dimensional themselves can be conveniently ignored for the purposes of the model.

We'll look at more esoteric enhancements of the 1DCA model later; for the moment we'll concentrate on a simple, fixed model.

Take a line of 28 cells. This line can conveniently be made equal to a single PRINT line on screen. Each cell can be thought of as a single character within that line, and its "state" - Alive or Dead - can be indicated by the letters A and D respectively. Later we'll look at the possibility of a cell being in other states (not necessarily having a straightforward description in human terms), and at representing states using graphics and colour - which is usually much more effective and makes it easier to spot patterns.

>>>MORE>

The state of any given cell at any given time is dependent upon its previous state and the previous states of one or more of its neighbours. YOU define the exact rules which govern the relationships between a cell and its neighbours, and YOU set up the initial states of a line of cells to which YOUR rules are going to be applied. The rules can be as simple or as complex as you wish, and we'll look at one example so that you can draw upon it for ideas.

In order that the rules be easy to apply, we'll make the line of cells exist in a "wraparound" Universe. If we didn't, we would need special rules to govern those cells which lie at the extremes - the first and the last - for they would each have only one neighbour, compared with all their neighbours, who would each have two.

In a wraparound Universe, the first and last cells are neighbours - if you need a mental model, think of the line as really a circle, the cells lying on the circumference, and the circle being cut and straightened out just for ease of viewing on screen.

Sticking to simple rules, we'll define rule number 1:

(1) At time T, the change in state of a given cell will be determined by the states at time T-1 of it and of its immediate neighbours.

There is an imaginary clock ticking away in this Universe, and it is represented by the variable T. Its units are of no importance, but if you need to call them something, then use the Stoidi terms of D'noces, Etunim, or Ru'Oh. On !Kreb, there is one D'noces to an Etunim, and one Etunim to a Ru'Oh. (As they say on !Kreb, give me a Hcni and I'll take an Elim.)

When T=0, or at "time zero", you set up the initial states of all the cells in the Universe, and display that Universe on screen. We'll see how shortly.

At the first tick of the clock (when T=1) you examine the state of each cell in turn, and determine its new state by taking into account the state of each neighbour either side. Once you have examined and updated all 28 cells, you can display the Universe.

The clock will then tick again, and you will go through the whole process again, displaying the Universe after each completed universal update.

For any given cell, its two neighbours will be either both alive, both dead, or one alive and the other dead:

```
A * A
A * D
D * A
D * D
```

where "*" signifies the position of the cell under examination.

>>MORE>

Now for some more rules:

(2) If, at time T, a given cell is ALIVE, it will remain alive if either of its neighbours was alive at time T-1.

(3) If, at time T, a given cell is ALIVE, it will die from overcrowding if BOTH neighbours were alive at time T-1.

(4) If, at time T, a given cell is ALIVE, it will die from loneliness if BOTH neighbours were dead at time T-1.

(5) If, at time T, a given cell is DEAD, it will stay dead unless BOTH neighbours were dead at time T-1.

Let's put these rules in a more easily understood form:

Rule 2

If the pattern is:

```
A A D  or D A A
  ^         ^
```

where the cell under examination is in the middle, then the cell will remain alive.

Rule 3

If the pattern is:

```
A A A
  ^
```

then the cell will die. (All together now, Ahhhh!)

Rule 4

If the pattern is:

```
D A D
  ^
```

then the cell will die.

>>MORE>

Rule 5

If the pattern is:

```
D D D
  ^
```

the cell will come alive; if the pattern is:

```
A D D   or   A D A   or   D D A
  ^            ^            ^
```

then the cell will stay dead.

Let's look at all the possible patterns, the rules which will apply to each, and the result on the state of the middle cell:

PATTERN	RULE	EFFECT
A A A	3	A -> D
A A D	2	A -> A
A D A	5	D -> D
A D D	5	D -> D
D A A	2	A -> A
D A D	4	A -> D
D D A	5	D -> D
D D D	5	D -> A

It might perhaps be stretching the imagination a little to try and explain, in human terms, how it is possible for a dead cell with two dead neighbours to suddenly regain life!

Once you have grasped the concepts of operation of this particularly bizarre Universe, you may feel confident enough to make up your own rules. My set here mean that the Universe will never cease to contain live cells for more than one tick of the clock, since a Universe filled with dead cells at time T will automatically have some come alive at time T+1 thanks to rule 5.

Equally, the Universe will never be totally filled with live cells for more than one tick of the clock thanks to rule 3.

The questions I have are these: is there a configuration which is stable from one tick to the next ? Is there a configuration which cycles round after a few ticks ? Is there an initial configuration which will always lead to a stable or cyclic Universe ? Is this set of

>>>MORE>

rules producing a cyclic Universe regardless of the initial state ?

I hasten to say that I don't have all the answers, so there is plenty of scope for you to participate.

Now to implementation. It is convenient to store the entire Universe (gulp) as a string variable, since groups of the letters A and D can be manipulated using SEG$(). You could of course use an array, with one element per cell.

For convenience in processing, it might be preferable to pre-process the Universe (!) by copying the status of the first cell and adding it to the end of the Universe (string), and copying the status of the last cell and sticking it before the beginning of the Universe (string).

That is,

```
                    (FIRST)-----(LAST)

                (LAST)(FIRST)-----(LAST)(FIRST)
```

This simplifies the implementation of "wraparound", since there is no requirement for checking for the start and end of the Universe for any special processing (see earlier).

In order to apply the rules, we need a table of the cases and of the results. In other words, the cases are AAA, ADA, etc., and the results are the effect upon the central cell in each case.

Note that when updating a cell's status, the original status of the cell (at time T-1) is not changed.

The new status is incorporated into the copy of the Universe being compiled (for time T).

If we call the string variable which holds the current (i.e. T-1) Universe, OLD$, and the string variable holding the updated Universe, NEW$, then things should become clearer.

Initial state

Determine an initial state for the whole Universe. You will need to define the state of each of the 28 cells. Using the letters A and D, the INPUT command suffices provided you validate the input to ensure (a) that only A and D are used and nothing else, and (b) that only 28 cells are entered.

You might choose to save time with special entries (like a null string to signify a Universe filled with dead cells), and that kind of future enhancement is up to you.

Assign the validated input to OLD$. Set time T=0, clear the screen, and display OLD$.

››MORE›

(An enhancement might be to display also the current value of T in some way. I leave the choice, and complications, to you).

Reset NEW$. This involves making it a null string, or emptying it ready for compilation.

You will need to have already created and filled a table of cases and results. I choose to use a two dimensional string array, with eight elements in each. The first dimension will hold the 8 triplets AAA to DDD for comparison with a triplet extracted from OLD$. The second dimension will hold the 8 single cells whose status has been determined by the first dimension. I call this array TABLE$(8,8), and I read the values which will become its contents from a series of DATA statements.

Thus:

ELEMENT	TABLE$(1,1 to 8)	TABLE$(2,1 to 8)
1	A A A	D
2	A A D	A
3	A D A	D
4	A D D	D
5	D A A	A
6	D A D	D
7	D D A	D
8	D D D	A

A loop will be used to run through OLD$, starting at position 1 and ending at position 28 (or at LEN(OLD$)-2, which is the same here).

A triplet will be extracted from OLD$ - we'll call it TRI$ - thus:

```
    FOR LOOP=1 TO 28
    TRI$=SEG$(OLD$,LOOP1,3)
```

A comparison will be made with elements of TABLE$(1,LOOP2) until a match is found:

```
    FOR LOOP2=1 TO 8
    IF TRI$=TABLE$(1,LOOP2) THEN EXIT
    NEXT LOOP2
    EXIT
```

(If the routine falls through the loop WITHOUT having found a match then a major error will have occurred with either the setting up of OLD$ or of the array. In addition, a FOR-NEXT loop has been shown here for simplicity's sake, but in the program listing I will use a "structured loop" to avoid problems with incorrect exit from LOOP2)

››MORE›

When a match is found, the corresponding element in TABLE$(2,LOOP2) holds the new status of the centre cell of the triplet, and that cell is then compiled into NEW$.

```
      EXIT:  NEW$=NEW$&TABLE$(2,LOOP2)
```

The use of the structured loop for LOOP2 avoids the creation of one of those nasty MEMORY FULL error conditions which can bedevil badly thought out routines.

The outer loop terminates with:

```
      NEXT LOOP1
```

completing the process for the entire Universe. Finally, a T=T+1 will move the universal clock on to the next "time frame".

NEW$ now holds a new Universe which contains 28 cells. You may choose to display it at this point, or to assign it to OLD$, reset NEW$, and THEN display it. The process can then begin again.

You can change the rules by changing the DATA statements, and even keep a sequential record of the Universe's existence by printing to paper or to disk, or even just storing in an array (OLD$() perhaps) for subsequent retrieval and analysis.

Simple Enhancements

The simplest enhancement is to exchange letters A and D for colours. As this model has only two states, we could redefine the letters A and D to be "filled" and "blank" - CALL CHAR(65,"FFFFFFFFFFFFFFFF") and CALL CHAR (68,"") - so that a choice of appropriate set colours for fore- and back- ground yield the two states; perhaps red and blue, or whatever.

At a later stage, this colouring process becomes more complex to implement, but not horrendously so.

The next enhancement might be to increase the number of states (increasing the number of cells makes display more difficult) that a given cell may experience. You might consider a "half-dead" state, or a "reproductive" state, or whatever. You can have as many states as you like, and note that your rules need not be highly specific as they are here.

A further enhancement is to increase the "neighbourhood effect". Either you can increase the number of neighbours having an effect on a cell (perhaps two either side, or three - bearing in mind that the processing and pre-processing of OLD$ will change), or be more selective, perhaps denying an effect for immediate neighbours and permitting only one or two distant neighbours to have an effect.

The example given later is "symmetrical"; you might choose to make one

>>>MORE>

left and two right hand neighbours have an effect, or to differentiate between, say, (DEAD)(CELL)(ALIVE) and (ALIVE)(CELL)(DEAD).

Here is the example listing, with some initial validation of input, and with a set of data conforming to the rules detailed earlier.

```
+----------------------------------------------------------------------+
| 100 CALL CLEAR                                                       |
| 110 FOR LOOP=1 TO 8                                                  |
| 120 READ TABLE$(1,LOOP),TABLE$(2,LOOP)                               |
| 130 NEXT LOOP                                                        |
| 140 DATA AAA,D                                                       |
| 150 DATA AAD,A                                                       |
| 160 DATA ADA,D                                                       |
| 170 DATA ADD,D                                                       |
| 180 DATA DAA,A                                                       |
| 190 DATA DAD,D                                                       |
| 200 DATA DDA,D                                                       |
| 210 DATA DDD,A                                                       |
| 220 INPUT "":OLD$                                                    |
| 230 IF LEN(OLD$)<>28 THEN 220                                        |
| 240 FOR LOOP=1 TO 28                                                 |
| 250 A$=SEG$(OLD$,LOOP,1)                                             |
| 260 IF (A$="A")+(A$="D")THEN 290                                     |
| 270 PRINT "ERROR: Wrong State: ";SEG$(OLD$,1,LOOP-1);">";A$;"<"      |
| 280 GOTO 220                                                         |
| 290 NEXT LOOP                                                        |
| 300 CALL CLEAR                                                       |
| 310 T=0                                                              |
| 320 PRINT OLD$;                                                      |
| 330 OLD$=SEG$(OLD$,28,1)&OLD$&SEG$(OLD$,1,1)                         |
| 340 FOR LOOP1=1 TO 28                                                |
| 350 TRI$=SEG$(OLD$,LOOP1,3)                                          |
| 360 LOOP2=1                                                          |
| 370 IF TRI$=TABLE$(1,LOOP2)THEN 400                                  |
| 380 LOOP2=LOOP2+1                                                    |
| 390 IF LOOP2<=8 THEN 370                                             |
| 400 NEW$=NEW$&TABLE$(2,LOOP2)                                        |
| 410 NEXT LOOP1                                                       |
| 420 T=T+1                                                            |
| 430 OLD$=NEW$                                                        |
| 440 NEW$=""                                                          |
| 450 GOTO 320                                                         |
+----------------------------------------------------------------------+
```

If anyone would like to submit rules/configurations for consideration, you might consider representing the 28 cell initial configuration as a group of 7 nybbles or hexadecimal digits. Thus an initial configuration of 28 alive cells could be represented compactly as FFFFFFF where a binary 1 indicates ALIVE and binary 0, DEAD. The rules can be easily indicated by writing the sequence of As and Ds as seen in the DATA statements above (DADDADDA) or again regarding ALIVE as 1 and DEAD as 0, so that a pair of hex digits encodes the rules (as here, 01001001 or 49 in hex).

We'll look at 2DCA in the next article.

>>>END

DRIVES & THE TI DISK CONTROLLER

A brief note about the capabilities of the TI disk controllers - both card and standalone.

Let's begin by dispelling a couple of myths: firstly, TI never produced any disk drives, just as they did not produce the TI Impact Dot Matrix printer. In both cases, "badge engineering" took place - a common practice, and not restricted to TI alone. The TI printer was made by EPSON and virtually Uncle Tom Cobbley And All made the disk drives!

Your drive may carry a TI badge, but under the covers it will bear a different legend: perhaps SHUGART, or MPI, or TEAC, or PANASONIC, or EPSON, or MITSUBISHI, or...

As far as both controller card (fitting inside the PEB) and standalone controller are concerned, as long as any drive conforms to a standard known as the SHUGART SA400L (which was a drive model number), then you should theoretically be able to fit and use it in your system.

However, there are differences. The standalone controller uses the same floppy disk controller chip as the card (the chip was made by a firm called WESTERN DIGITAL, but it is obsolete and has been for a number of years!) but does not possess the same circuitry to support that chip.

Unless you modify your standalone controller (see the ITUG project in TI-LINES V3 issue 10, with a few errors picked up and published in issue 11) it cannot use DOUBLE-SIDED disk drives as other than SINGLE-sided. The controller card CAN use double-sided drives without modification.

The DISK MANAGER I module can intialise single-sided drives but NOT double-sided, although later software - DISK MANAGER II and other better (in some respects) packages - can do so.

Early single-sided drives were only capable of using 35 tracks, and could therefore make use of only 315 sectors, while 40 track drives could use 360 sectors. Double-sided drives are usually also 40 track as a minimum, permitting 720 sectors per disk using the TI disk controller (card or modified standalone). Software other than DISK MANAGER I can format such disks.

To date, the TI disk controllers are restricted to SINGLE DENSITY data format. A PROM set is being offered in CANADA which apparently provides double density storage, even though the chip mentioned earlier has no double density pin. I am advised that TI have incorporated a special circuit somewhere within the controller card - known as an external data separator - but I am not able to confirm if this is so, for the time being at least. If true, perhaps that is how the Canadian PROMs provide the increased density.

I am advised that the DENSITY capability is a function of the controller and not of the drive. If a drive is advertised as being double density, this does NOT mean that you can make use of that capability when using the TI disk controllers.

››MORE›

Some drives have either a fixed 80 track capacity or can be switched between 40 and 80. I have yet to do some further practical experiments, but so far it would appear that PRACTICAL use of 80 tracks is beyond the TI disk controllers as far as DOUBLE-SIDED use is concerned.

However, information furnished by RICHARD BLANDEN and confirmed by COLIN HINSON indicates that the DISK MANAGER II can initialise an 80 track disk in an 80 track drive BUT - only as SINGLE-SIDED. This provides the same amount of disk space as would be obtained from initialising a 40 track double-sided disk (i.e., 720 sectors). Attempts to initialise 80 track double-sided inevitably end with only half the disk space being allocated, and the second side does not appear to be properly formatted.

The Canadian PROM set apparently does permit full 80 track usage.

You CAN use 80 track drives as double-sided units, provided you only initialise to 40 tracks. However, this confers no real advantage as no-one else can read your disks (unless they too have 80 track drives).

You cannot read "normal" disks recorded to 40 tracks with an 80 track drive unless you possess another, third party, controller, or unless you have written special software to double-step the read/write heads.

You CAN fit half height drives within the PEB - in fact, you can fit 1/3 or 2/3 height drives instead - but you may experience problems with the supply of sufficient power if you do not beef up the PEB power supply or if you cannot (or do not) implement "motor on select" on the drives, if available. Motor on select simply means that only the drive about to be used has its motor turned on; under normal operating conditions, the TI disk controller switches ALL drive motors on when ANY drive is accessed.

You can have up to 3 disk drives in use with the TI controllers, and I am advised that four 8 inch drives could be substituted if you wanted to really put yourself out on a limb...

The TI disk controller card has TWO connections on it. The two are very different, but provided the correct counterpart connectors are used, no difficulties should be encountered. Both connectors may be attached to drives, but the total may not exceed three at any one time.

TI state in their documentation that the final drive in a chain MUST have a "terminating resistor pack" in order for the controller to function correctly. The resistor pack indicates electronically to the controller which of the connected drives is the "last". However, in some cases we have found that a few drives require ALL to have such terminator packs before the system will function.

A chain of drives is regarded as those drives connected by the same piece of ribbon cable to the controller. As there are two connectors on the card, there are potentially two chains. If one connector has two drives attached to it, and the other connector has the remaining drive attached to it, then the "end" drive of the pair (the last on the cable) AND the single drive on its own MUST have terminator packs in order to conform to specification. However, this is prone to variation (!).

Drives can be configured - usually on their circuit boards - to be a specific drive number, usually 1 to 4 (or 0 to 3). This does not permit us to use more than 3 drives at any one time. The configuration may be

>>MORE>

achieved either through "jumpers" or "sleeves" or through a "strapping pack". On numerous models, both the strapping pack and the terminator resistor pack's socket are found close together, near the edge of the circuit board where the ribbon cable's edge connector fits on.

In the early stages, TI provided a special cable with at least three connectors attached to it - together with small pieces of circuit board for two of the connectors, with each piece being reversible but having a slot cut in it to fit a "key" in a specific connector - and this does appear to have caused some problems for almost everyone. The concept behind the use of such a cable was simple: it enables ALL drives to be configured as Drive 1 but lets the ribbon cable determine which drive will be called 1, 2, or 3 by the controller.

It began to cause problems when people obtained drives which had been already configured as drive 2 or 3.

By and large, you may be better served if you replace the TI version of the cable with a standard unit, and have your drives reconfigured as you desire.

Some drives will automatically start their motor when a disk is either inserted or removed from them; this is not a fault, but a benefit and can reduce the probability of a disk being erroneously clamped inside the unit. Very few drives of this type have been seen in TI systems.

In rare instances, TI (or other supplier) provided double-sided drives instead of single-sided, and in at least two cases, 4A owners were blissfully unaware of this. Considering the difference in cost of the two types of drive, this action is rather difficult to understand.

However, in at least one case the double-sided drive had been physically altered electronically to turn it into an apparent "pair" of drives - that is, side 1 of the drive was regarded by the controller as drive 1, and the other side as Drive 2. This produces disks which can only have their second side read by a similarly-configured system!

Such drives were commonly used in Acorn (BBC) systems and in the RML380Z (Research Machines Ltd) and possibly others, and unless a drive has been SPECIFICALLY designed for use with one of those systems, it should be possible to reconfigure it for "normal" use on our system.

I am advised that the format used by IBM for the electrical connection of their drives may not be compatible with our SA400L standard. Until I receive confirmation one way or the other, I would remain wary of the purchase of such drives. I have been advised that the TI recording format is taken from an IBM standard, but to date I have been unable to confirm this, especially since our format is 256 bytes per sector, 9 sectors per track, while the IBM "standard" varies considerably but is a minimum of 512 bytes per sector, 8 sectors per track (increasing through 9, 10, and 15 sectors per track).

It is possible, using machine language software, to directly access the TI disk controller chip and to record data in all sorts of peculiar formats - BUT - at the end of the day the sums must add up to provide the equivalent of 256 bytes per sector, 9 sectors per track.

A number of other drive types are available, notably the 3 inch used by

>>>MORE>

AMSTRAD, and 3.25 and 3.5 inch (the former a Hungarian speciality, I am told). As some of these do conform to the SA400L standard, they CAN be used in the 4A system but, as with 80 track drives without a true 80 track controller, you would be placing yourself out on a limb.

The sectors on a disk formatted by a standard TI controller are laid out in a specific sequence. They are not numbered 0, 1, 2, 3, 4, 5, 6, 7, 8 in sequence, but are placed in the positions which allow most efficient retrieval/recording. I do not have the exact details to hand at the time of writing, but a subsequent article taken from an overseas magazine will provide full details.

This non-sequential layout of sectors is called the INTERLACE, and on rare occasions, if reading a disk which has been formatted using a third party controller with a different interlacing (required when the speed of access to the drive is changed) you may experience an audible pause if the sound of the head stepping can be heard above the PEB fan...

This is simply because the interlace used is not efficient when used with the slower TI controller (track to track access time around 20ms) and not because there is a fault with the drive. Other controllers can increase the speed of access to around 3 ms.

Almost any type of disk can be used with a given drive, except the very high density disks which require a higher current in the read/write head in order to make a lasting magnetic impression on the disk surface. If you do not have a drive specifically designated as high density, do not make use of such disks if your data is very valuable to you.

It is possible to make use of single-sided, single density disks in a double-sided drive and initialise them to 720 sectors.

However, the manufacturing process apparently begins by producing a sheet of disk material and then testing it to see exactly how much data it is capable of storing reliably on a given surface area. If one side of the sheet proves even a little unreliable, the material is classed as single-sided. Further tests elicit its capability of storing data in 80 or 40 track, single or double density formats.

It is not recommended that you use disks which fall below the minimum specification for the application you are using.

Finally, some 80 track drives were manufactured with 40 track heads - the "width" of the read/write head produced a recording path which was too wide for 80 track use. These drives are invariably switchable, but cannot be used in practical terms in the 80 track mode, because the heads, when stepped at 80 track intervals, produce 40 track wide paths which inevitably overlap and corrupt each other when recording.

This information is taken from a variety of sources, including personal experience over the last four years. I make no rigid claims as to its accuracy, but to the best of my belief none of the statements above is grossly in error.

If readers do soot something which they know to be wrong, I would be very grateful if you would contact me (Ol' Baldie) and let me know in as much detail as possible, so that everyone will ultimately benefit.

››END

MACHINE CODE = FOLLOW UP

In the article MACHINE CODE in V4.2, I commented on PROGRAM FORMAT files and said that I didn't know why that format wasn't universally used by TI machine code programmers, since it was the most compact and the fastest to load.

A little further reading around the subject has elicited the answer.

Yes, program format (or MEMORY IMAGE) files DO load faster, but the files are inflexible. Why and how ? Because program format, unlike TAGGED OBJECT CODE files which are DISPLAY FIXED 80, are non-relocatable which can be a handicap.

So what's so great about being relocatable ?

A relocatable program is one whose format enables it to be placed at any location in memory and run without difficulty. An example in BASIC would be most informative - except our BASIC (and probably others for all I know) doesn't permit relocation! I'll have a go anyway.

Imagine that you have a form of BASIC which allows you to load more than one program at a time into memory. (Please don't write in about MERGE format files in Extended BASIC - they are not comparable). The first problem to overcome is what to do when all the programs begin at line 100. The first program would load correctly, but any subsequent program would overwrite or displace the one currently in memory - you can't have two line 100s, two 110s, etc. (Unless you muck about using CALL LOAD() to change line numbers, but that won't necessarily make two such programs run correctly!).

The solution is to have the computer keep track of all the program line numbers, and alter any subsequent programs' line numbers accordingly.

So, if the first program loaded in (we'll call it program A) occupies lines 100 to 2000 inclusive, then the next program (call it B) will have to be loaded and changed during loading so that it occupies lines AFTER 2000. If B happens to occupy lines 100 to 1770, then ALL of its line numbers (and line number references) will have to be changed - relocated - by having 2000 added. This will make B occupy lines 2100 to 3770.

If a third program, C, which normally would occupy 100 to 1430, is loaded, it will have to be relocated so that it occupies lines 3870 to 5200. Note that the increments (usually 10) will be 100 at the points at which each program "begins" in the listing.

Things are never easy, though. Any GOTO, GOSUB, ON GOTO, ON GOSUB, or IF..THEN..ELSE occurring within either B or C will also need to be relocated by the appropriate amount, and if A, B, or C refer to each other (as in GOSUB "C" or IF X=Y THEN "B") then those jump destinations will need to be adjusted.

On top of this, any variables will need to be manipulated. This is

>>>MORE>

perhaps easier in machine code than in BASIC, since in machine code you would not need to "rename" the variables to avoid conflict in their use (unless passing contents from one program to another).

Bear in mind that all of this is having to be done during the process of loading the program into memory - a fairly enormous task.

Now, in BASIC terms, you might not see the need for having more than one program resident in memory at one time. In machine code terms, the need can arise frequently - especially when using languages like C99 or Forth, or making use of the excellent FWEB loaders.

In addition, (on our system) all machine code programs are "named", and when resident in memory, in order to be recognised as existing, their names and current memory locations are stored in memory as well (in what is known as the REF/DEF table). Our BASIC programs are not "named" in that sense (unlike other BASICs) and only when stored on disk is a name usable (but only for storage purposes).

A relocatable program, then, is more flexible than a non-relocatable one (usually program format, but not always since DF80 can also be non-relocatable unless specified to be otherwise!).

The program which actually does the loading is capable of coping with the relocation process provided the format conforms to what it expects. This is where terms like "absolute addressing" and "indexed addressing" apply. Programs which use absolute addresses are inflexible - indeed, there is the infamous case where Atari (I believe) produced a number of cartridges with programs on board which used absolute addresses. The result was that the cartridges would only work on some, but not all, consoles.

What are absolute and indexed addresses, and why would they result in programs working on some consoles and not others ?

An absolute address is in fact any address which is specified by its location. Thus a command to jump to memory location 1000 or to store a value in memory location 2000 is using an absolute address.

An indexed address is a different animal. Instead of instructing the computer to retrieve a value from a specific address in memory, you refer instead to a table of addresses. Your "address" refers to an entry in the table, which contains the address you want. Sometimes the entry might be not an address but another table, with even further tables being connected to it. The address you need might in fact be several tables away. The use of tables means that the final addresses can be changed over and over again in different consoles; as long as the location of the entry leading to the address isn't changed, then any software can be used on any console without difficulty. The Atari software used values which were found at one location in some consoles but which existed at different locations in other consoles. If the programmers had used TI's technique of always using tables to access data, they would have fared better.

As you are now thoroughly confused, you may rest for a month...

››END

REPAIRING YOUR TI POWER SUPPLY UNIT

By JOHN ROE

The TI power supply unit is a pretty robust affair and nothing much seems to go wrong with it. However, the cable connecting the PSU to the console (the output cable) on my machine has broken three times.

It seems that while this cable is adequate for the electrical load placed upon it, it is not mechanically strong enough to withstand the strain of unpacking for use and packing away again after use, which I was doing for some time. I now have a permanent layout set up so I no longer have this problem. As other TI-LINES readers may have had similar trouble I have set out below my experiences in performing the necessary repairs. For the sake of completeness I have also included suggestions for dealing with other types of PSU failure.

There is a warning notice on the side of the TI PSU to the effect that there are no User serviceable parts inside. Take no notice. This is a typical TI con. Nearly all faults can be easily put right by anyone able to use a screwdriver and soldering iron. It is therefore quite in order to remove the top cover of the PSU by unscrewing the screws in the base. All you will find in the expensive looking box is a stepdown transformer with two fuses protecting the primary and secondary windings. There are no other electrical components.

You will also need a continuity tester, either bought or home-made. This usually consists of a suitable battery and bulb combination with probes to contact the terminals of the equipment to be tested. When contact is made if there is no break in continuity the bulb lights. If you have a multimeter use the resistance setting.

The TI PSU is not really a power supply unit as I understand the term. That is a unit which supplies a voltage (or a choice of voltages) suitably rectified and smoothed to produce a steady DC current for use in otherwise battery powered equipment. The TI unit only steps down the voltage which is still AC. The rectifying and smoothing circuitry is contained inside the console. This is perhaps a fortunate thing for the amateur electrician who finds that he is not getting any power through to his console, as it simplifies the fault finding process.

First of all, the obvious: check your mains supply and then check the fuse in the mains plug using your continuity tester. Having then removed the cover of the PSU check these fuses. They can be replaced by their equivalents from Tandy's or similar supplier. The required sizes are indicated on the printed circuit board next to the fuse holder; 0.2A for the primary winding and 0.5A for the secondary winding. While you're using the continuity tester you may as well at the same time test each of the windings for continuity although it is extremely unlikely that there will be any fault here especially if the fuses are still intact.

If all fuses and the windings are OK the next thing is to check the input and output cables for continuity. This is where the problems can

››MORE›

arise. A break can often be diagnosed by simply touching or moving the cable while the PSU is connected and the console switched on. Even a slight touch will sometimes cause the broken ends to make contact momentarily and thus switching on the machine. If this is the case, you can, with care, locate the break fairly accurately. If the break is somewhere along the length of the cable and not too near either end it is a simple matter to repair the break either by soldering or by using some sort of joint and covering the repair with insulating tape.

If the break is too near the end to allow a simple joint to be made then you may have problems. The input (mains) cable is not too difficult.
If the break is near the plug end, just cut the cable at the break and rewire the plug to the new end. If the break is near the PSU end then you will have to cut the cable at the break again, and unscrew the cable clamp inside the box so that you can get at the terminals easily to resolder the new ends to them. The two mains wires are Blue or Black (phase neutral) and Red or Brown (live). The Red/Brown wire should be soldered to the terminal protected by the 0.2A fuse.

Similar action can be taken if there is a break in the output (to the console) cable too near the PSU to make a joint, but as there are three wires you will have to note which colour wire goes to which terminal to make sure that you resolder the new ends to the correct terminals. In my PSU the colours are Red, Black, and White, but they may be a little different in others. This is of no importance as long as you replace the new ends correctly.

It is when the break occurs at the console end of the output cable that you are really in trouble. TI in their wisdom used a moulded plug.

This consists of a hard blue plastic inner plug containing the pins to which the cable ends are soldered, and an extremely tough black plastic outer cover which has been moulded on to the inner plug carrying the pins and soldered connections. There is no way that this can be disassembled by an amateur electrician, at least not so that it can be put together again. It is quite impossible to remove the outer cover cleanly to expose the wiring and pins. After a considerable struggle I eventually managed to remove the outer cover but not before in the process the wires had parted company from the pins to which they were soldered, and I was suddenly faced with the problem of deciding which wire went to which pin. Not to worry, I thought. The pins are numbered on the face of the inner plug as you look at it from the console end, thus:

```
+--------------------+
|                    |
| 1               2  |
|      o      o      |
|                    |
|                    |
|    o          o    |
| 3               4  |
|                    |
+--------------------+
```

(the plug goes into the console upside down). Now I knew that I had seen something somewhere which would tell me which terminal was connected to which pin. I spent hours searching through

>>>MORE>

all the back numbers of TI-LINES, TI*MES, the Parco Magazine and the Home Computer Magazine, without success. If you find yourself in the same position, don't waste your time looking through back numbers. Just turn the PSU upside down and there it is, boldly embossed in the plastic base of the box.

Output pins 1, 2 16V~ /1.6A
Pins 2, 4 8V~ /0.15A

The only problem then was to identify which of the PSU terminals should be wired to which pin. The TI PSU contains a stepdown transformer which changes our 240V mains supply to a nominal 24V. On my PSU however this actually measures out at a little over 27V. The console, however, does not use this 27V supply. Instead the secondary winding is tapped a third of the way along its length, and the voltage between this tapping and the nearer end measures 9V+ and between the tapping and the further end is 18V+. Both these voltages are used by different parts of the computer, so you will find three terminals on the secondary winding with three wires soldered to them. There are places for four pins on the inner plug, but there is no pin at number 3 position.

Now to identify which of pins 1, 2, and 4 of the plug are to be connected to which wires of the cable. With the PSU connected to the mains, use the AC voltage setting of a multimeter to check the voltages obtainable at the ends of the cable. If the cable ends are labelled A, B, and C, then use the probes to contact A and B, then A and C, and then B and C. One of these pairs should give you a reading of the order of 24V. Then the end which was NOT used for that reading must be wired to pin 2. The end which gave a nominal reading of 16V (together with the end already wired to pin 2) should be wired to pin 1. This leaves the remaining terminal to be wired to pin 4. If you ever find yourself faced with rewiring the console the console plug I would recommend that you replace the complete cable at the same time using a mechanically stronger cable to avoid the same trouble on future occasions.

Lastly, what if your tests reveal that a transformer winding has gone ?

The short answer is to buy another PSU. If someone has a u/s console he wants to get rid of, you could pick it up at a reasonable cost, possibly free. Failing that, another working console complete with PSU could be picked up fairly cheaply perhaps. Otherwise, you can buy transformers from shops like Tandy's. Anything you are likely to buy won't of course be a good mechanical fit inside your present PSU box but you could also get some sort of a plastic box from the same supplier and wire in a couple of fuse holders and the connecting cables.

From the electrical point of view you need to consider first the power handling capacity of the transformer. As it is fused the primary winding of a TI transformer won't handle power greater than 50W, but it is better to allow a safety factor of, say, 20%. The secondary winding as fused cannot handle more than at most 15W. Again, allow a safety factor of 20%. Then you need a stepdown ratio such that you get a nominal 24V across the two ends of the secondary winding. As indicated by my measurements, you can allow 10% or so up on that. There must also be an intermediate tapping of the secondary winding at a point 1/3 of the way from one end of the winding.

››END

S T R U C T U R E D   L O O P S

An explanation and discussion of the so-called "structured" loop

If you have been reading around the subject of computer programming, you may have encountered structured loops, or at least come across a mention of them somewhere (even in this august issue!).

What are they, and why are they supposed to be so marvellous ?

To begin to understand the advantage of structured loops, we'll first have to identify and discuss unstructured loops (!). This is fairly easy, since we only have one example in TI BASIC (or Extended BASIC for that matter): the old standby, FOR-NEXT, or to give him his full title, FOR-TO-STEP-NEXT.

If you've not yet grasped how to use FOR-NEXT, or not understood the concept of looping, now is your chance (but not your only one).

There are many occasions on which a programmer might need to have recourse to a loop. The loop is a very simple concept to grasp. Loops are used whenever a section of program (maybe even the entire program!) needs to be executed more than once. For example, if a delay in part of a routine is required, the simplest solution is to go "round and round" a few - or a few hundred - times:

```
1000 FOR DELAY = 1 TO 2000
1010 NEXT DELAY
```

This type of loop is often called a FOR-NEXT loop since the BASIC words FOR and NEXT mark the beginning and end of the loop within the listing.

When a section of a program needs to be executed a known number of times - even if the number varies, and has to be supplied as a variable:

```
FOR J = K TO L STEP M

One or more BASIC statements

NEXT J
```

then a FOR-NEXT loop is a very useful tool.

```
100 FOR LOOP = 65 TO 90
110 PRINT CHR$(LOOP);
120 NEXT LOOP
```

››MORE›

In the example on the previous page, the variable called LOOP is used to print out the letters of the alphabet in upper case. It can be useful sometimes to use a variable name which indicates that variable's purpose as long as it doesn't conflict with a reserved word!

You may notice that the letter "I" is used as a variable name in an inordinate number of cases of published listings. This is a hangover from the days when loops could only be executed using just one variable, whose name was I. As today's amateur programmers pick up their habits - good and bad - from examples of programming from the past, the use of I has been perpetuated. Like STOP, END, and the human appendix, it is something which used to have an essential function but is now no longer so vital.

The problems with using FOR-NEXT begin if you ever need to jump out of the looping program segment:

```
1000 FOR J = 1 TO 15
1010
 --
 --  program segment
 --
1090
1100 IF X=Y THEN 150
1110
 --
 --  program segment
 --
2000
2010 NEXT J
```

In the above (contrived) example, an exit from the loop is made if the contents of variables X and Y are equal. This is potentially a disastrous manoeuvre.

When a FOR instruction is encountered, the computer sets up the "conditions" for beginning and ending the loop. Until the NEXT part of the loop receives the value stipulated as the "end" of the loop (in the example above, the ending value for J is 15, so 15 - or a value greater than 15 - will cause the loop to finish), the computer keeps the loop "active".

This means that if your routine subsequently causes line 1000 to be executed again, the computer will set up ANOTHER set of beginning and ending conditions, still keeping the first version of the loop in existence. Each setting up uses up memory, and if the process of exiting and re-starting is repeated often enough, the computer will run out of memory and crash with a MEMORY FULL error.

The correct way to exit a FOR-NEXT loop is to supply the NEXT with its correct exiting value, and the computer will then "drop" the loop from memory.

However, in practice it is not always convenient to implement such a correct exit, since you may want to make use of the current value of the variable controlling the loop, although if your routine jumps out

››MORE›

of the loop and back in again (!) without setting up a fresh FOR, then such problems may never arise. Don't show a computer science lecturer such a program, though!

The better approach is to set up a so-called structured loop. This implementation does not use FOR-NEXT, but relies instead on IF-THEN and GOTO. It requires a little careful thought in setting up, but it does have the advantage that you can exit and restart as many times as you like without using up further valuable memory.

FOR-NEXT operates using the "do-until" approach - that is, when NEXT is encountered, the associated variable is altered (depending on what value - if any - is specified by STEP, if used; the default is 1) and THEN a test is made to see if the variable exceeds the specified ending value. If you enter this:

```
    100 FOR Z = 1 TO 10
    110 PRINT Z;
    120 NEXT Z
    130 PRINT Z
```

you'll find that Z is 11 when the loop exits correctly, because of this.

Using IF-THEN and GOTO, a much tighter control over the loop can be exerted. This is the structured version of the previous example:

```
    100 Z = 1
    110 PRINT Z;
    120 Z=Z+1
    130 IF Z<=10 THEN 110
    140 PRINT Z
```

The starting value for Z is set up in line 100. The increment of Z occurs in 120, and the test is in 130. Now try this:

```
    100 Z = 1
    110 PRINT Z;
    120 IF Z>= 10 THEN 150
    130 Z=Z+1
    140 GOTO 110
    150 PRINT Z
```

The position of the increment and test, and even of the type of test (<, <=, >, >=, =, <>) is very important. This is why it requires extra care in implementation.

Because such care has to be exercised, you won't often see such loops in listings; frankly, it can be too much like hard work.

However, if you are concerned that your programs should be as error-free as possible, you should take note of the use of the structured approach to looping.

>>END

TI BASIC : LONG LINE ENTRY

Entering extra-long lines in TI BASIC

New subscriber DAVE SMITH from Leamington Spa alerted me to an omission in my irregular series for Beginners. I did briefly cover the subject in V1.1 of TI-LINES, but haven't touched on it since. I did think that I'd covered it in my dreaded book, but apparently not, which is odd, since the facility of entering longer lines is one I came across a very long time ago.

What exactly am I talking about ? The facility of entering TI BASIC statements which extend beyond the four screen-line limit set by TI.

Dave encountered the problem when attempting to type in a published program listing. He found a statement (a DATA statement as it happens) which he could not complete, because the cursor would not travel beyond the four line limit and allow him to continue to the end of the statement as published.

The apparent limit is 4 x 28 = 112 characters per TI BASIC statement, but a little cunning (aided by some knowledge of what goes on "behind the scenes") can extend this to around 6 lines (6 x 28 = 168). The actual limit is less than 168 characters, but the true value escapes me for the moment. (In fact, if you type in 1 REM press the space bar once to give a space after the REM, and then press and hold down CTRL then press U and keep it pressed until the end of the fourth line is reached, and then, puff, puff, press ENTER, you will have entered a very, very long - and useless - line indeed. You don't think so ? Try LISTing it and see...).

You can obtain more detailed information by reading the CONTROL AND FUNCTION KEYS articles in volume 1, and anything on TOKENS, but here are the essentials:

1. Nothing is what it seems. Commands like PRINT, INPUT, CALL, GOTO, etc., actually occupy a lot less space than you think, BUT...only AFTER you have pressed ENTER. Once you have typed (correctly) a BASIC statement and then pressed ENTER, the computer's inbuilt program replaces "reserved words" with single characters, called TOKENS, so that even a large word like RANDOMIZE eventually occupies a space equivalent to a single character. When you tell the computer to LIST your program, another inbuilt program translates these single characters back into the long words (like SEQUENTIAL) which you originally typed.

2. To take advantage of this shortening effect, you need to enter any extra-long lines in two (or more) stages, to give the computer a chance to compress each stage and thus allow more space on screen for entry of further BASIC instructions - achieved through EDITING. If you want a wacky analogy, consider that most modern dustcarts stop every so often to allow huge compressors inside to squash the garbage into a smaller volume, and thus permit more dustbins to be emptied before needing to

›››MORE›

make a trip to the tip.

3. Pitfalls. Even with the technique described below, you may not always be able to enter as long a line as you would like, and you may have to restructure the line into two or more shorter lines. As a general rule, though, if the line has been published as part of a printed TI listing, then you should be able to enter it, too.

However, you must exercise a little care when deciding where to "make the break", or the computer's internal checks will issue an error report (* INCORRECT STATEMENT or whatever) and foul everything up - nothing will have been accepted.

4. What to do. Choose a suitable point at which to press ENTER on the entry of the first part of the line. This is where 99% of the problems may arise. If you have not balanced the parentheses (equal numbers of open and close brackets!) - if there are any - or completed a reserved word - e.g. RANDOM instead of RANDOMIZE - then everything may well be in vain and you will have to begin entering the line all over again.

You may even have to enter what amounts to a wrong - but correct - line. For example, if the end part of a line reads something like this:

```
  ...A*(B+C)/D-E/F+G^H-I*J+K
```

and you can only fit ...A*(B+ in, then the simplest thing to do is to enter a "valid" statement - for example, change the "+" to ")" so that you now have ...A*(B). The computer accepts this because it satisfies TI BASIC's rules. Now call up the line for editing. If you have been using the automatic line numbering facility (NUM or NUMBER at the start) you will have to exit it with either ENTER, FCTN 4 (BREAK), or something similar (i.e. FCTN E or FCTN X).

Either type EDIT and the relevant line number, or type the line number and press FCTN E or FCTN X. If all has gone well, you should find that you can run the cursor (using FCTN D) beyond the A*(B). In that case, you are in a position to change the ")" into "+" and continue entering the rest of the line.

In rare cases, you may have failed to fool the machine entirely, and you might just still be unable to extend the line. At this point you need to turn to a specialist for help (Editor buffs his nails) or you may decide to soldier on. What may have happened is that your chosen break point is too short of the four line limit, and the computer may duly oblige by sticking to four lines still. You might get round this by temporarily turning the line into a REM but I don't guarantee it.

By and large, the majority of extra-long lines pose no major problems and you should experience no difficulty. Note that the operative word here is "should", which means that Soddes Law could well apply. In that event, don't give up. Drop me (or someone) a line, or use the phone, or go banging on someone's door (at a reasonable hour!).

Have fun!

))END

N O T I C E   B O A R D

WANTED / 4 SALE / WANTED / 4 SALE / WANTED / 4 SALE / WANTED / 4 SALE /

BRIAN BURKE is selling some TI gear. It consists of a boxed TI-99/4A console, the modules PARSEC, THE ATTACK, MUNCHMAN, EXTENDED BASIC plus manual, MINI-MEMORY plus manual and line-by-line assembler, VIDEO CHESS, SPEECH SYNTHESIZER, and software STARTER PACK 1 and 2, GAMES WRITER 1 and 2, and several tapes with games etc.

He is asking £75 for the lot, and you should contact him at 43 Station Road, Warboys, Huntingdon, Cambs., PE17 2TH.

I am looking out for THERMAL PAPER for the TI THERMAL PRINTER on behalf of a few desperate owners. If any ITUGer knows of a source of suitable paper would they please contact me as soon as they can.

I have a lone DOUBLE-SIDED 40 TRACK FULL HEIGHT MPI drive secondhand, suitable for insertion in the PEB or into an external case with power supply. The seller is asking £60 including post and packing for a quick sale, and would prefer cash. Contact me in the first instance.

DAVE HEWITT has experienced a problem with C99 and has asked me to put out a request for assistance - this was the only spot I could fit it in! He finds that any attempt to open three files simultaneously (as when compiling from one disk file to another, and sending the reports to a third) causes an error. Contact him on 0865 341428 if you can help.

I have a small selection of hardware and software items being sold for other Users through me. The size of the list and its contents is fluctuating continually, so I cannot reliably report here what it will consist of. There should be at least one PEB, 32K card, Extended BASIC, MiniMemory, Video Chess, Adventure, Wumpus, PRK, and a number of other items including commercial tape games and books by PECKHAM, INMAN/ZAMORA /ALBRECHT on BASIC, and WATT on LOGO (not TI-specific). Again, it has to be a case of first come, first served.

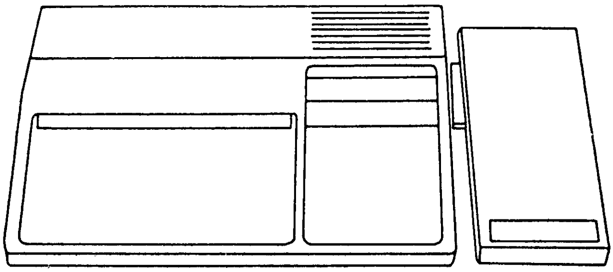
I have mislaid the details of the ITUGer who expressed an interest in a pair of half-height drives. Would you please contact me again ?

I can still supply disks (double-sided, double density, 96tpi format but can be used single-sided, single density, 40 track without problems) at only £7 per pack of ten. No library cases, but the price is inclusive of post and packing.

I still have one Extended BASIC module and one MiniMemory module, both at £27 each inclusive of post and packing.

Over the next few pages you will see an advertising feature which covers the products which ITUG intends to supply over the next year. Some of the equipment comes from West Germany, and is subject to 9% import duty plus 15% VAT, making approximately a 25% surcharge. The prices reflect this, and also include the cost of post, packing, and insurance.

>>>MORE>



ADVERTISING FEATURE

The daisy-chaining of standalone peripherals by TI has in the past received a considerable degree of criticism and was discredited by TI themselves with the introduction of the more compact and efficient - but more expensive - peripheral expansion system.

However, the standalone peripheral concept never died completely, and lately has experienced something of a revival, partly because of convenience for owners of unexpanded systems, but largely due to the scarcity, and continued high cost, of the peripheral expansion system.

In the special case of one product listed here - the MECHATRONIC 80 column standalone card - the ONLY way it can provide its impressive graphics capability is as a standalone; there is insufficient space within the console to allow it to reside there, and the timing and signal strength considerations apparently rule out implementation as a card for the peripheral expansion box.

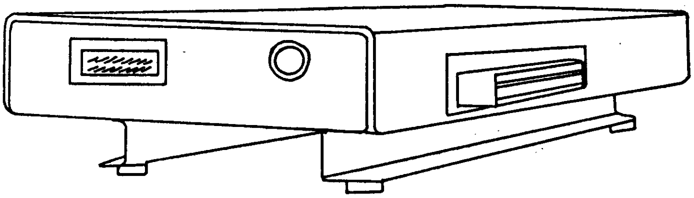
Over the next few pages you will read brief, outline, details of the products to be supplied through ITUG. The prices would have been highly competitive had it not been for a 9% import duty coupled with a further 15% VAT, adding at least 25% to the prices before the cost of post, packing, and insurance were added.

Because ITUG does not have the financial cushion of a large market and multiple retail outlets, please ensure that you are committed to your order; if you changed your mind because you weren't entirely sure that you wanted an item, ITUG could be left in an awkward situation.

Also included is a review of the MECHATRONIC 80 COLUMN STANDALONE CARD by MACK McCORMICK which should provide some details on its potential.

If you decide to make an order, please state clearly what you want, and the address to which it is to be sent, and make cheques payable to ITUG.

>>>MORE>



MECHATRONIC 128K RAMDISK WITH CENTRONICS PRINTER INTERFACE £105.00

Designed as a standalone, with data lines passing through the unit so that a PEB may be connected, enabling owners of expanded and unexpanded systems alike to make use of it.

It comprises 32K standard User RAM (configured by the 99/4A's operating system (O/S) into the usual 8K for machine code utilities and 24K BASIC program area); 96K RAM configured as a RAMDISK (actually a little larger than the equivalent 40 track single-sided single density disk); and a Centronics printer interface.

The full 128K can be bank-switched in 32K banks by your own software (provided that software resides outside the 128K space - for example, in the MiniMemory) and there is a 24 page booklet covering this and other aspects.

The RAMDISK O/S will currently only permit a maximum of THREE files to reside within the 96K space - we are investigating the possibility of altering the O/S to make it directly compatible with the TI Disk O/S format (127 files).

The Centronics port is not the same as TI's PIO (which is parallel, but does not conform to the Centronics standard) and avoids name conflict by being called PLOT. This means that if you already own an RS232/PIO card there will be no signal conflict.

MECHATRONIC 80 COLUMN STANDALONE CARD £235.00

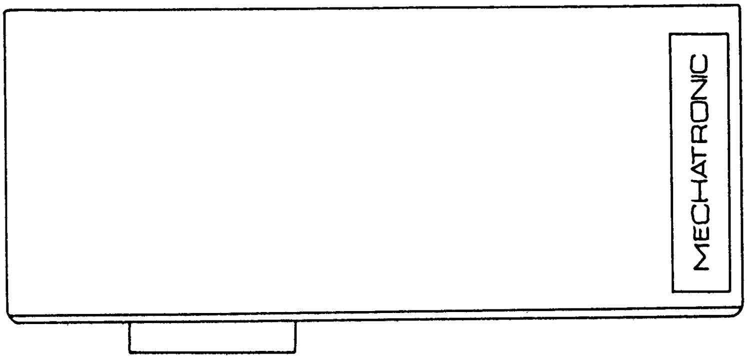
This product received a highly complimentary review (A+ overall) from MACK McCORMICK, and precis of that review is published here.

Ease Of Use: A   Documentation: B   Performance: A   Value: A

Overall Rating: A+

The Mechatronic 80 column card is the final answer for the TI-99/4A.

>>>MORE>



In fact, if you already have a RAVE 99 keyboard and a GRAMKRACKER or GRAM card, you already have the equivalent of the new Myarc 9640 micro. My system running this configuration and the 9640 O/S (which executes perfectly from an Editor/Assembler cartridge with RAM at >6000) was only about 5% slower than the 9640. This is because much of the 9640's speed derives from the 9938 VDP chip which is also at the heart of the Mechatronic 80 column card.

Ease of use. The card is transparent except when running standard BASIC when a micro switch on the card must be pressed so the O/S can find the extra 128K of VDP RAM provided. Installation requires the removal of the old TI VDP chip from within your console - if you are at all dubious about performing the replacement, ITUG can arrange for the work to be done for a nominal fee. Usually the VDP chip is socketed, so work with a soldering iron is unnecessary. Mechatronic provide pictorial step by step instructions. Total installation time is about ten minutes.

To fully benefit from this card you really need an RGB monitor, since a standard TV may not be able to cope with the high resolution graphics.

Documentation. The manual is short but adequate. All DIP switches are fully documented and the RGB output connections are clear. The manual does a good job of documenting how you get 80 column output from within your current TI programs by printing to the card, which is treated like a peripheral.

The company provide 80 column versions of TI-Writer and MultiPlan with each purchase. (The MultiPlan was demonstrated at the recent BLOXWICH WORKSHOP).

Performance. The card performs exactly as expected, is well behaved, and I have never had it crash a program. With the extra 128K RAM you can run the 9938 chip in all modes including super high resolution bit map mode. Converting Extended BASIC programs to give displays in 80 columns is a snap. You simply print to the card using a special print sequence which also specifies colours and other necessary information. From TI-Writer you can also see your print in 80 columns on screen before printing it.

>>>MORE>

The improvement in screen resolution and being able to have the TI work in 80 column mode is fantastic. This is the last piece of hardware you need to upgrade your machine to the equivalent of many other newer systems on the market.

Overall rating. It is rare for me to assign an overall grade of A+ to a product, but this one is well deserved. An important side benefit is that software developed for the Myarc 9640 should be compatible with this card. For my money the 80 column card from Mechatronic cannot be beaten.

The 9938 VDP specifications are extensive and too complex to provide full detail here; it ranges from 40 and 80 column modes with 512 colours to 512 by 424 pixels with 16 colours. The total palette of colours ranges between 1024 and 4096, depending on the information source.

MECHATRONIC MOUSE £118.00

This product comes with its own power supply and connects via the joystick port. It received an airing at the BLOXWICH WORKSHOP recently, and comes complete with a game of BREAKOUT and patches to enable its use with TI-ARTIST. There is also a demonstration version of TIDOS (see elsewhere).

The source code is provided with full instructions for adding to your own software, so that assembly language programmers can make alterations to suit.

Mice (or Mouses ?) have begun to appear with a number of recent micro packages (e.g., Atari 520ST, MYARC 9640), and they are useful when coupled with the correct application programs (i.e., for selecting from complex on-screen menus where key-pressing could become slow and far too involved).

MECHATRONIC SUPER EXTENDED BASIC II PLUS £47.00

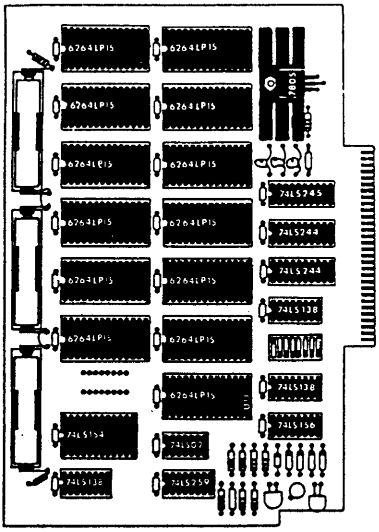
Recently reviewed by PETER WALKER in TI*MES, this comprises a module and a disk for expanded system owners (with additional graphics utilities).

Mong its commands are BHCOPY which dumps text screens to EPSON compatible printers, MSAVE which saves part of CPU RAM to disk or cassette (and MLOAD which brings it back on board again), QUITOF/QUITON which disables/re-enables the QUIT key, and FIND which searches an array for a specified string and returns the element number at which the string was found.

It is also possible to transfer copies of blocks of memory from CPU RAM to VDP RAM and vice versa, enabling screens to be prepared and swapped at speed. Menus could be switched in and out rapidly using this command and thus speed up certain types of program.

If you are into graphics then the additional graphics commands, which provide facilities which should have been contained as standard within TI BASIC let alone Extended BASIC, will enable you to experiment to your heart's content.

››MORE›



TI99/4A INTERN

Heiner Martin

The Operating System of TI 99/4A Internal ROM and GROM Listing with Commentary and Directions for GPL

vth

Verlag für Technik und Handwerk GmbH

NEW HORIZONS RAMDISK   BARE BOARD + O/S   £50.00
READY BUILT SINGLE-SIDED EQUIVALENT (96K) * £170.00
READY BUILT DOUBLE-SIDED EQUIVALENT (192K) * £200.00

---

*The prices for the ready built versions of the NEW HORIZONS RAMDISK are subject to fluctuation due to the changing prices of the main HM6264 RAM chips used.

This RAMDISK is a battery-backed card to fit in the PEB. It comes with its own O/S based on DM1000 and the RAMDISK can be configured so that it emulates a particular drive (for example, drive 1. You could load your most-used software onto it with an appropriate LOAD program heading a selection procedure, then set the RAMDISK to emulate drive 1, so that when using Extended BASIC, selection of XB automatically loads and runs your software - say, FWEB). A full specification was published in an earlier issue of TI-LINES (V2.12) and another advertisement should appear next issue.

AN UPGRADE IS AVAILABLE - CALLED V6.3, FROM MIAMI USERS - AND UPGRADES THE CARD TO 256K WITH AN IMPROVED O/S. THIS IS A SEPARATE PACKAGE AND IS AVAILABLE THROUGH WEST MIDLANDS TI USERS. PRICE ON APPLICATION. RING 0922 476373 FOR FULL DETAILS.

OTHER PACKAGES AVAILABLE

TI-99/4A INTERN BOOK by HEINER MARTIN. This book gives details of the TI O/S and GPL disassemblies. Price £10.00 inclusive.

TIDOS - disk and manual - giving a true disk operating system. £25.00

>>>END