TIHOME

tidings

VOLUME 2 , NUMBER 2 MORDEN SURREY 28th APRIL 1982

TIHOME is not affiliated with Texas Instruments in any way and is supported only by its subscribers.

sTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIding
dingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsT

In this edition:-

TIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdingsTIdin

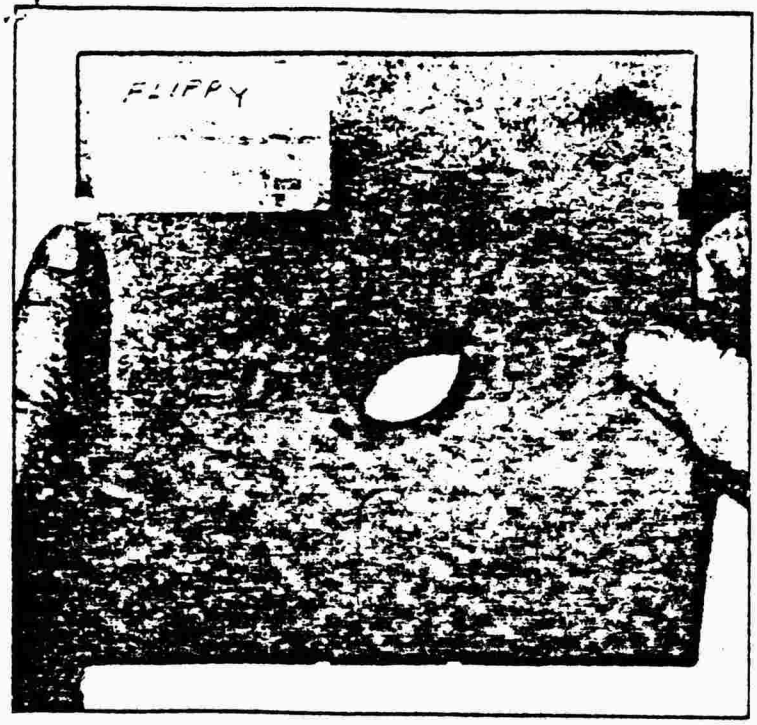
EDITORIAL

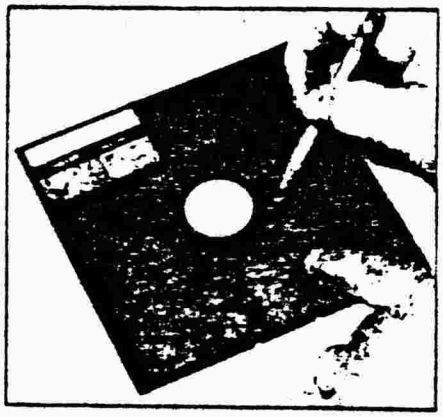
TIHOME continues to do quite well. Membership is around the 180 mark and still rising. Gradually I am getting it organised and slowly turning it into a more and more professional body. Membership seems to increase by leaps, usually at the beginning of each month, I wonder why!

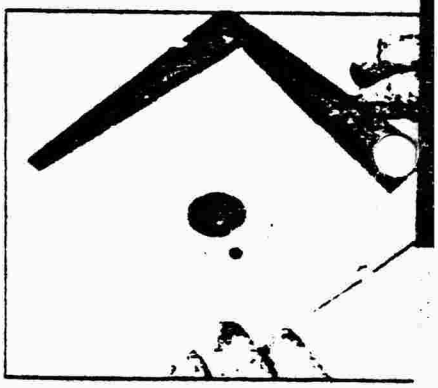
The content and format of Tidings has remained virtually unchanged over the last 18 months. With the increase in membership, the content and format could do with looking at. Therefore, I have included a questionnaire in this edition and I would be grateful if you would fill it in and return it as soon as possible. I need to know what you would like to see in Tidings. Remember, if you don't ask, you don't get. While we are on the subject of Tidings, please remember that I am always pleased to receive items for inclusion in Tidings. Don't worry that it might not look very professional. All offerings and tips are gratefully received. Someone, somewhere, is trying to work out a problem you have just solved; so let the rest of us in on the secret.

TIHOME has received quite a number of programs from Holland for inclusion in the Software Catalogue. Unfortunately, they are in Dutch. If anyone is willing to do a translation job, please ring me at any time on 01.640-7503.

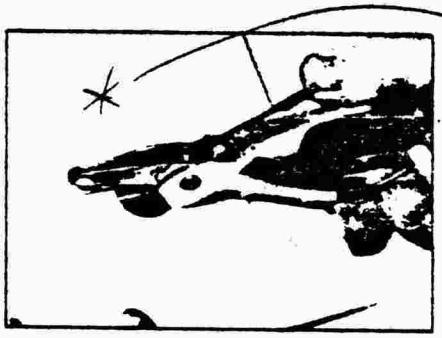
Please note the question in the questionnaire about the use of your name and address. Personal contact is something I would like to encourage, but not without each member's permission.







Various photos showing the transformation of a standard, single-sided floppy disk into a double-sided floppy with twice the storage capacity. Note that the template must be used, and that all the holes must be punched in the proper locations with the utmost accuracy. So be careful.



* THIS TYPE OF HOLE PUNCH IS BEST!!

MAKING FLOPPIES INTO FLIPPIES

Double Storage Capacity Of Floppy Disks

By Herb Friedman

ALMOST FROM THE BEGINNING of personal computing, we've been warned not to try and use the *flip side* of standard 5-inch single-sided diskettes. And, in the beginning, there was good reason for this warning: single-sided disks are *certified* only on one side; and pressure pad tracking against the flip side of the disk (opposite the head) might pick up loose oxide chips and grind them into *both* sides when the disk was used.

But technology has moved along. The modern high-quality disks from brands such as Scotch and Maxell (among others) actually have two usable surfaces, although only one is certified for single-sided pricing. Also, interior liners within the jackets are very good at picking up and holding loose chips and flakes long before they get to the pressure pad. Finally, quality disks are made very well, and they simply don't flake or shed. So there's really no good reason why you shouldn't try to get more storage for less money by flipping the disk—turning a *floppy* into a *flippy*.

In fact, one of the finest drives, the Siemens FDD-100-5—which is used by Heath, among others, is preassembled for optional write protect and sector index sensors that permit flippy operation of unmodified single-sided disks. (This option is available only to OEMs and do-it-yourselfers; don't ask Heath or any other manufacturer to do it if their basic drive isn't a *flippy*.)

Double Your Capacity. Being able to use the flip side doubles the storage capacity. If your disk system provides 102K per disk (one side), two sides provide at least 204K. It's like getting an extra disk for free.

Since very few drives can be electrically modified for flippy use by adding optional sensors, the easiest way for the hobbyist to get flippy operation is to punch an extra Write-Enable Notch and Index Window in existing disks to permit the drive's normal sensors to function on the *flip* side, when the disk is turned over.

Indexing And Lockout. Take out one of your 5-inch disks so you can inspect it carefully. Note that one edge has a Write-Enable Notch. Within the drive is a mechanical switch or photo-optical sensor that determines when the notch is open. If the notch is open, the computer can write to the disk, and record or erase data. If the notch is covered by a small piece of tape, the switch or optical detector is locked out, thereby locking out the WRITE mode. No data can

2021 note: Sadly the rather flippant assurance that all will be well, given above, was not well made -whilst the warnings against the practice proved well made- user groups that placed all their libraries onto flippies were unable to later transfer them. There was a large failure rate of floppies kept in damp garages and lofts but even well kept flippies had a VERY high failure rate. bb

Make use of the flip side of your disks to double data storage capacity

be recorded nor can any file be erased. The drive will only READ from the disk.

Located near the Central Window —the large hole in the center of the disk—is a small hole called an Index Window punched in the jacket. If you carefully rotate the disk itself—the magnetic media—you will see a small hole appear within the Index Window. You might have to rotate the disk almost a full 360 degrees before the small hole appears because soft-sectored disks such as used in the Radio Shack TRS-80 have only one hole in the magnetic media.

Hard sectored disks have either 10+1 (11) or 16+1 (17) holes depending on the particular computer, and a hole will appear almost as soon as you rotate the disk.

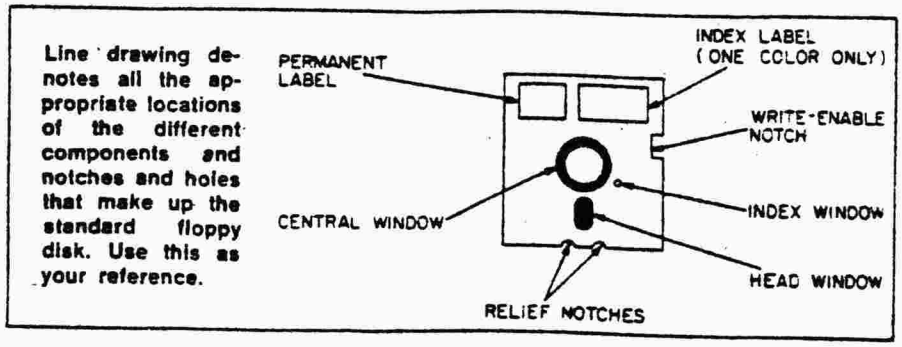
The hole provides the sector information to the computer through a photo-optical sensing beam that passes through the hole.

To make a floppy disk into a flippy disk, all you need do is punch an additional WRITE ENABLE notch and Index Window so the sensors work when the disk is flipped. To see how this works out, hold your disk in the normal loading position and note the positions of the notch and window. Now flip the disk over. Note that the notch and window are no longer in the correct locations.

Unfortunately, you simply can't take a paper punch and punch two holes where you think they belong. The disk isn't all that fussy about location, but the notch and window must be well inside the ballpark. But that's easy enough to do with a paper punch and a homebrew template.

Making the Template. The template is made from a file folder. Spread the folder open, lay the disk on the folder and carefully trace the outline, the notch, and the Central Window.

Rotate the magnetic disk so a hole is centered in the Index Window. Position the disk on the template, carefully matching the location of the notch, and mark the location of the Sector Hole (through the Index Window) on the template. Put the diskette in a safe place and then cut out the template (but not the notch) and the center hole (within the existing Central Window).



Line drawing denotes all the appropriate locations of the different components and notches and holes that make up the standard floppy disk. Use this as your reference.

On the left you see the template which is used to locate the exact center of the floppy disk. Once the center is certain, then the procedure of determining the location of the hole which is used for optical guidance may begin. Using the square cardboard template, punch out the hole as shown in the right photo.

From the scrap that's left, cut a strip about 1 inch wide and 8 inches long. Carefully punch out the notch from the template. Then reach in with the punch through the center hole and carefully punch out a hole at the Sector Index mark. This completes the template.

Position the template on the diskette so the Index Windows and Write Enable notches are in perfect alignment. Then flip the template.

Using a pencil or fiber pen, mark the jacket with the location of the template's notch and Index Window. Then carefully shift the template to the opposite side of the disk and mark the location of the Index Window on the opposite side of jacket (make certain the notch remains in the same position). Set the template aside; you won't need it.

Punch out the new notch in the jacket (you'll have to squeeze hard). Then lay the disk flat and slide the small strip you cut from scrap between the jacket and the disk directly behind an Index Window mark. Slide the punch between the jacket and the strip, center the punch over the mark, and punch out one side of the new Index Window. Make certain the "punching" falls out.

If you punched the Index Window correctly you should be able to rotate the magnetic media itself until a sector hole appears, and you should be able to see right through the diskette. If you can't see through you have made a punching error.

Now try the flip side of the disk in your drive. If you have a TRS-80 make a backup of TRSDOS on the flippy side. TRSDOS verifies all tracks and locks out defective tracks so you can get a good idea of how successful your flippy will be. Heath computers have a separate media check used for locating defective tracks (if any). CP/M users can try for a *verified* DUPE. If your computer has some other way to check for defective tracks (or media), use it. (WHC NEED CPM ::)

If you're using high quality disks, it's more than likely that every flippy will give 100 percent satisfaction. Older disks and budget priced disks might be another story. If you see too many tracks being locked out, it's best not to use the disk as a flippy. ■

* USE THE HOLE PUNCH TO CUT THE NEW WRITE ENABLE NOTCH AS WELL!

TEMPLATE

TEMPLATE

Babbling Brooks

By Pete Brooks

I don't quite know what happened to the last issue of Tidings; apart from the late distribution (outside Paul Dicks' control), the subscription reminders for all, (lucky me, I had TWO copies, with and without reminders), the omission of Quotes From Your Letters, etc., some kind person re-typed the first page of Babbling Brooks (BB from henceforth), carefully missing out two lines of text with the explanation of what the listings were about, and taking great pains not to include the relational operators in the listings.(Not to mention the new calendar month that was created, although, to be fair, they did correct my mis-spelling of 'waffle').

This month's BB therefore begins with a quick post mortem on last month's, plus details of that special deliberate mistake (ahem!) which of course you all spotted, and so did I, but only three weeks after I'd submitted the article, and Paul Dicks was supposed to get busy with the Tiprex...

Anyway, the missing lines on the first page of BB were an intro as part of the extract from Peter Phillips' letter (the quotation marks weren't supposed to be all the way up the page like that...) and should have appeared just before the listing thus:-

"Pete Brooks' article in Tidings on Incremental statements gave me a challenge I couldn't resist, so here is a short program which will bounce an asterisk up and down the screen:....."

A little further down the page, and line 135's statement should have read:

```
135  R = R + (R>24) * (R - 24) + (R<1) * (R - 1)
```

The 'deliberate' mistake was towards the bottom of the 12th page in the article, and should have read Adjacent / Hypotenuse = Cosine , NOT Adjacent / Opposite. I have been carrying a lead brick around in my handbag for a week as penance. (If you believe that, you'll believe anything.)

I have since seen a copy of the official TI handout on the extra CALLs provided by the PRK and STATISTICS modules, and I'm not floundering around in my own ignorance as much as I was. Even so, not all the details have been given, and I await further enlightenment. If you possess either module and you are interested in learning more, I strongly suggest you buy a copy of the handout from Paul Dicks. There are enough extra facilities to warrant a whole section of the library being devoted to programs which use the extra Calls. The president of the Netherlands' User Group, Paul Karis, is likely to produce an article and a program in the not-too-distant future which will enable creation/modification of PRK and TI Basic+PRK files, interchangeably.

This month's BB should also herald a change in structure, as I hope to divide this month's, and future BB's, into sections dealing with a wide range of topics, provided that it isn't too ambitious, or too egocentric. For example, in additiion to covering a main topic each month (this time a continuation of the plotting facilities given last issue: two further degrees of resolution, and the provision of the DRAW command as another subroutine), I want to provide an insight into the advances taking place in other fields, mostly as a result of the application of the kind of technology in which you have invested. This section will cover things of scientific origin, and if you have anything to add, by all means write in. This month's item is a little goodie called the Replicator, and it is straight out of science fiction. That should have whetted your appetites a little.

Other items will include NMR (Nuclear Magnetic Resonance), a topic which I should be able to explain in some detail as one of the research doctors at the hospital in which I work is pioneering the technique of examining minute areas of the body tissue without the need for surgery, and which will undoubtedly in the years to come REPLACE surgery in its traditional form; Flat screen television, several of which are now entering the marketing stage, and in which Britain, through 'Uncle' Clive Sinclair of ZX80/81 fame, is leading the field; Space science - with details of the staggering plans currently at large, mostly in the USA, and which are past the 'what if' stage, and are now at the 'when can we finish building it' point; and items concerning microcomputers - what the next steps are likely to be.

If any of you feel sufficiently moved to write on any of those topics, or indeed on any other, even if what you have to say fits on just one A4 sheet, get your pens out, blow the cobwebs off your Olivetti, and hammer out a few lines for the benefit of the rest of us. One warning, though: it can be habit-forming. You may find yourself spending hours and hours composing short pieces on every subject under the sun; like Mass Drivers, Terraforming, Microrefrigeration, and Josephson Junctions. .Need I say more ?

All that this is intended to do is to infect you with the same enthusiasm for the Age ahead, and to help make you aware of the immense number of possibilities whic. are about to be created, not forgetting their consequences. The gap which exists between us and say, a nomadic Tuareg tribesman, is going to increase at a phenomenal rate. In order that that gap should not prove a liability, in particular for the disadvantaged, it is imperative that we, not impersonal government bodies, not self-interested commercial concerns, but we, as self-educating members of a sophisticated (!) society, should be aware of, and capable of responding to, the needs of others who are less aware (for whatever reason) or less able to take advantage of the opportunities ahead.

And that reminds me, one section of BB is going to be a Soapbox, in which you or I will pontificate (at least, I will pontificate, you, if you write in, will say things that are interesting and relevant). And I think that's enough pontificating for now; on with the show.

Last month's BB should have taken some of its own medicine and numbered the listings given; if you have any queries, write to me at:-

68, Kelburne Road
Cowley
Oxford OX4 3SH

No telephone, although I can sometimes be reached at work on Oxford 64811, extension 335. If the answering machine is on, PLEASE leave a message.

This month's BB is being written in some haste and is bound to generate some problems, so I apologise in advance for any omissions, re-typings, and general verbosity.

One point which has come to my notice thanks to member Dickon Lush is that 4A's have a bug in the CALL KEY command which prevents the correct operation of a couple of the listings given last issue. When using the 'User-directed' plotting routine, although pressing Down (shift X on a 4, but a function key I believe on a 4A) has no effect, even though K does appear to contain the value 0, I am informed by Stephen Shaw that the value returned is in fact non-zero, but so small that it is displayed as zero. Use of INT(K) = 0 should cure the problem; in fact it might be an idea to insert K = INT(K) before testing as in the listing. I have written to Mike Lunch of TI about this, and if he writes back I will pass on any information he may have. Alternatively you may find a page from TI in this issue of the newsletter, and they may deal with it there. The bug with the printer is apparently a universal one, as Paul Karis tells me, and TI may have decided not to correct it, as so many units are in circulation.

Some issues back (Volume 1 No. 4) I presented the first article which detailed how 99 owners could implement the equivalent of AND or OR. Further work has shown that the OR given is in fact INOR or Inclusive OR. Its counterpart is EXOR, or Exclusive OR, which can also be implemented. If that sounds like gobbledygook to you, especially newcomers, then it might help you to think in these terms:

AND requires that both items being tested (e.g. IF A=1 AND B=1 THEN...) should BOTH be TRUE in order for the line number following THEN to be jumped to.

INOR requires that EITHER ONE OR THE OTHER OR BOTH be TRUE for the same reason.

EXOR, however, requires EITHER ONE OR THE OTHER BUT NOT BOTH to be TRUE.

The equivalents are implemented using Relational Expressions and Numeric Operators (specifically "+", "*", and "-") thus:

```
AND is   (one expression) * (another expression)
INOR is  (one expression) + (another expression)
EXOR is  (one expression) - (another expression)
```

AND and INOR can be used for several expressions as once, but EXOR can only be used for pairs of expressions. If you want further details to be given, write in and say so.

Just out of interest, NOT can be implemented on single expressions by using 1 + (expression): however, it is usually easier to "invert" the expression: i.e., if it is (K=0), then although NOT(K=0) can be 1 + (K=0), it is usual if the operating system permits it, to use (K<>0), and so on.

I seem to be suffering from printing glitches everywhere these days: I wrote a letter to Personal Computer World a couple of months back, and they published this month — carefully omitting a NOT which turned everything that I'd said on its head. Makes you wonder whether it's worth bothering, doesn't it.

I had also intended to include a short routine to enable the plots given last time to be "inclined"; that is, tilted through a specified angle; however, I've managed to temporarily mislay my notes and listing, so that will be left for next time.

This month's main topic continues from last month's. The 192 x 256 resolution plotting subroutine can be modified to produce a 96 x 128 resolution plot, where each "point" is composed of 2 x 2 of the higher resolution dots. The algorithm changes slightly, as the number of possibilities of combinations of hexadecimal numbers is reduced: i.e., as each point is 2 x 2 dots in size, a) each plotted point will modify TWO hex numbers instead of just one, and b) the only hex digits to be used will be 0, 3, C, and F. The new algorithm can operate slightly faster because of this: it is simply a matter of determining which of the four hex digits is defining the present plotted point, and what the new digit will be once the point is plotted.

The resolution can be reduced still further by another factor of 2, giving a plot area of 48 x 64. This is the size of the plotted points which are used by Sinclair's ZX81 for example, and at this level the algorithm changes completely. The size of each plotted point means that there are only four positions available within each 8 x 8 character; this reduces the number of shape variations to a manageable 16, and so they can be predefined instead of requiring redefinition each time. In comparison, the 96 x 128 plot has 16 possible positions available within each 8 x 8 character, giving rise to 65, 536 possible variations, whereas the 192 x 256 level has 64 positions within each character, giving rise to a staggering 1.845 x 10^19 possible variations. The lowest level of resolution likely to be of any use is 24 x 32, where only one position occurs within each 8 x 8 character, and only two variations are possible (i.e. black or white squares). The number of possible

variations is linked, surprise, surprise, to binary maths. To find out how many variations a particular degree of resolution produces (by variations I mean the number of different possible definition strings which would be required), first work out how many positions within the 8 × 8 matrix of a character a given point could occupy. Take two to the power of this number, and you have the total number of variations. Why two? Because each point plottable can be either plotted or unplotted — two states. So for example, the number of 8 × 8 sized points you can fit into an 8 × x dot matrix is 1, and 2 to the power 1 is 2: i.e. either the point is "on" (FFFFFFFFFFFFFFFF) or "off" (0000000000000000). With the Medium II level plot (given later), each point is 4 × 4 dots in size, or a quarter of an 8 × 8 dot matrix character, so you can get 4 such points within the matrix (N.B,, without overlapping). 2 to the power 4 is 16 (2 × 2 × 2 × 2). Figure 1 (see listings at the end of the article) shows the 16 possible variations, with the first character (binary 0000) showing the four "quadrants" in which plots can be made. The shapes may appear on the surface to be arranged at random, but in fact they follow the by now familiar 0000, 0001, 0010, 0011, ... 1110, 1111 of 0 to 15 in binary. The first character is entirely empty, except for the quadrant markers, which is equivalent to 0000; the second has a point plotted in the 4th quadrant, which is equivalent to 0001; and so on. (Read from left to right, top to bottom). Their definition strings range from 0000000000000000, 000000000F0F0F0F, ... through to FFFFFFFFFFFFFFFF.

The Medium I plotter uses a 2 × 2 dot point, and you can fit 16 of these into one 8 × 8 dot matrix character, without overlapping, so the number of variations is 2 to the power of 16, which is 65,536 (see earlier). The highest level of resolution (given last issue) has 64 possible positions in an 8 × 8 dot matrix (because each point is one of the dots rather than a group of them), and 2 to the power 64 is a vast number of variations: around 20,000,000,000,000,000,000!

As usual, there will be a fairly detailed explanation of how the routines work. But first, the modifications to the high resolution routine, and an explanation of how the DRAW subroutine works.

On the same page as Fig. 1 there is a listing of the modified plotter together with the DRAW subroutine. (The DRAW subroutine appears by itself for clarity on the next page of listings.) The change that has been made in the initialization affects line 150. During a conversation with Howie Rowe, another Oxford 99 buff, I tried explaining how the conversion from hex to binary and back again had been effected. Howie pointed out that it would have been simpler to have placed "spacing" field separators (I used full stops) in the hexadecimal reference string in line 150 so that the position of the first digit of each four-digit binary number in the binary reference string corresponded to the same position for the corresponding hex digit. Followed that? So that's why there are all those dots in between the hex numbers in line 150. This means that all the hours spent typing out the reasoning behind the mathematical conversion from one to the other which appeared in the last issue, has gone up the chute, as you don't need to convert. This also means that lines 1170 and 1190 will also need to be modified, as they perform the conversions in either direction. If you have the last issue of Tidings and you are interested in plotting pretty pictures, then take note of the revised lines given in the listing.

I managed to find the equations necessary to produce the DRAW subroutine in a book called Documenta Geigy, a treasure house of all sorts of information, but alas, not normally available to the layman, I believe, In fact, it is very hard to get hold of a copy: the issue I perused is several years old, and is the prized possession of our Medical Library. The equation used is one that permits you to "extrapolate" (that is, project a line in the same, or opposite, direction from the points that you currently know) given two points, particularly on a graph.

The equation is:

```
y=y1 + dy/dx   *  (x -x1)
```

and its counterpart is:

```
x=x1 + dx/dy * (y-y1)
```

Now, before your eyes start to glaze over, and your head begins to loll, let me say that things are NOT as difficult as they may at first appear. All you need to follow the equations are the keys to understand what all the little letters and symbols mean. You don't need to be a genius to understand equations like these, and I hope to provide a sort of maths "primer" in the episodes of BB to come, so that those who thought they had left things like this back at school can ease themselves into the symbol-filled world of maths with only a few hiccups.

The first thing to note is that in order to avoid confusion over the letter "x" and the multiplication sign "×", I have used the asterisk, as in Basic, to denote multiplication. I don't have the Greek letter "delta" on my typewriter — it looks a little like a triangle (Δ) — so I have used the lower case letter "d" instead. In mathematical parlance, "d" followed by another letter like "x" or "y" or whatever, stands for "the change in" whatever the value of the letter may be. Basic is very strongly linked to algebra, so that the translation between Basic and algebra is very easy. For example, if we let "H" stand for your height, and four years ago it was 6 feet (all right, about 183 cm then), but now it's 10 feet (305 cm), then "dH" is the change in your height, which is four feet (actually, it is "plus" four feet; you could just as easily have shrunk four feet). We would call the first measurement of your height at 6 feet, H1, and the second measurement at 10 feet, H2. The change in your height is then dH = H2 - H1. In those equations above, the "y" values correspond to screen row (like R in the subroutine), and the "x" values to the columns (C in the subroutine). It's just unfortunate that I chose to use X and Y in the subroutine as well, but they don't have much bearing on the present discussion.

The two locations on screen between which we want to draw a continuous straight line are x1, y1 and x2, y2. dx is x2 - x1 and dy is y2 - y1. Give any value for y or x lying on that straight line, you can find the corresponding x or y. So if we ran loops from x1 to x2 or y1 to y2, those equations would give us all the necessary information to plot all the points between our two screen locations.

There is just one slight problem. Suppose that our line is either directly vertical or horizontal? That means that the values for one set of co-ordinates wouldn't change. (i.e., if vertical, only the row value would change; the column value would stay the same). This would make one of the "d" values zero; under certain circumstances, you could find the program crashing because it has attempted to divide a number by zero.We need to make allowances for that occurrence by making it a special case to be tested for, and acted on differently. Also, what happens if there is no change in either position? That is, that the two points on the screen coincide? We need to make a special case for that one as well. And what happens when we come to draw a line that is spanning many columns, but only drops two or three rows in the process? If we loop according to the rows, we may only get two or three points plotted unless we perform some fairly nifty maths to cope with it. However, if we find out which of the two planes (rows or columns) changes by the largest amount, that should ensure that we get a continuous line. If the change for both "d" values is the same, it doesn't really matter which one we use to control the loop.

There is one final problem. If the values that we choose to control our loop with run from a large value to a small value, instead of the other way about, how do we cope with that? Fortunately, there is a function in many Basic dialects which helps. It is the SIGNUM or SIGN function: SGN in TI Basic. The SGN or a number is +1 if the number is positive, 0 if the number is zero, and -1 if the number is negative. The standard increment on a loop is +1 (the STEP value) unless you specify otherwise. Using SGN you can set the computer to take care of that side of things for you. The "d" values tell you whether the loop should have a positive or negative STEP value, although an attempt to STEP 0 will cause a crash, so that is another reason for testing for a "d" value of zero (signifying no change).

The tests boil down to these:

1) If either dx or dy is zero, a separate set of routines need to be used.
2) The program needs to use the larger of the two "d" values as a loop control.
3) If there is no change in either position (dx and dy are both zero) there is no plot to be made.
4) If only one of the "d" values is zero, you need two routines to cope with whichever one isn't zero.

In the listing I have used X1, Y1, and X2, Y2, as well as DX and DY, so that you should have no trouble equating the symbols. R and C are equivalent to y and x, which might cause some head-scratching. L is used as the loop control value. I have perhaps wrongly, left in an extra line which could have been taken out. Line 850 should have read (DX = 0) * (DY = 0) THEN 780 (on the grounds that as a simple RETURN needs to be executed, the jump will operate faster if it has only a short way to go: the 99s seem to begin searching for a jump destination by beginning the search at the first Basic line every time) i.e., the use of the AND equivalent — see earlier. That would mean that line 860 was unnecessary. However, I just HAD to give an example of EXOR somewhere in the listing, and that's were it came.

It gives you something to think about anyway.

I have also used the trick of using INT and 0.5 to round up/down as described in the last article. There is one further thing which some of you may not be familiar with, and this is the ABS function. ABS is usually short for ABSOLUTE, and it gives the positive value of whatever number is used with it. For example, ABS(-12.3) is +12.3 (but without the "+"); ABS(.7) is still .7. It is used in line 780 for the reason that we are testing for the larger of two values representing change, rather than testing to see which of the two numbers is greater. If that sounds odd to you, look at it this way: if the change in column value is -12, and the change in row value is +3, then -12 is less than +3. But we are not testing for that, we are looking for the actual size of the change, and 12 is a bigger change than 3. To go back to the height analogy, someone whose height has increased by 3 inches is taller than someone of the same original height who has shrunk by 12 inches, but the chap who has shrunk has experienced the greater change in height. Maybe someone sawed his legs off...

To use the DRAW subroutine, all you need to do is set values for X1, Y1 and X2, Y2 and GOSUB 690. The subroutine takes care of the rest, except of course that it has no validity checks in it, so yo will have to make sure that it doesn't try to plot off the edges of the screen. Don't forget that you need the plotting subroutine in 1000 onwards, and the initialization at 100 onwards, whichever level of resolution you are using.

For the masochists, a line by line explanation.

690 Works out the change in X.
700 Works out the change in Y.

710 If either the change in X, (DX), or the change in Y, (DY), or BOTH, is zero, (INOR, remember ?) then jump to the routines set aside for those special cases.

720 When it comes to deciding which value to use as a loop control, I plumped for DX only when it is less than or equal to DY as a first choice. This loop uses DY. ABS was explained earlier.

730 Because of the tests just performed, any DY value here MUST be non-zero; therefore the loop can safely be used with SGN(DY) as a step value without causing any problems. If Y2 happens to be smaller than Y1 (i.e., DY is negative), then the step value of SGN(DY) will be -1.

740 As our loop control value, L, is being used with Y values (rows), the value for R (to be passed to the plotting subroutine at 1000) is simply the rounded integer value of L.

750 As Y values have been chosen as the point of reference, in order to find what the corresponding X value on the same slope would be we have to use the second of the two equations given earlier. L is our reference value for 'y' and so it replaces it in the equation here. DX / DY is the slope, and the .5 is the rounding value. (Note that with this system, you don't need to round the X1, Y1, X2, Y2 values before passing them to the 690 subroutine).

760 Having determined the R and C values, we can now plot them.

770 End of the loop.

780 Once this loop has been completed, there are no further calculations to be made, and so we can RETURN to the main routine that called the DRAW subroutine.

790 This is the parallel subsection to the one we have just been through. If the degree or magnitude of change on the X axis had been greater than on the Y, a jump would have been made from line 720 to here. Again, the value for DX MUST be non-zero, as we have screened out any zero values in line 710. This time, L, the loop control value, runs from X1 to X2 with a step value determined by SGN(DX).

800 This time, because the loop is being used with X values (columns), the value for C is simply the rounded integer value of L.

810 This time also, because X values have been chosen as the point of reference, we use the first equation (note that the slope is DY / DX this time) to work out the corresponding values for R.

820 Make the plot.

830 End of the loop.

840 Return to the main routine.

850 This line was jumped to from 710 if either or both of the 'd' values was zero. As explained earlier, this line would be better as an AND so that if both 'd' values were zero, no plots would need to be made, and so we could RETURN immediately by jumping to the earliest occurrence of a line with RETURN in it. (In Extended Basic of course you could use RETURN instead of a jump destination). If you decide to replace the EXOR equivalent here by an AND equivalent, the line will become IF (DX = 0) * (DY = 0) THEN earliest occurrence of RETURN, and line 860 will be defunct, so delete it. What line 850 tests for is one or other BUT NOT BOTH of the 'd' values being zero, and if so, jumps to 870; if BOTH are zero it continues to line 860 which causes a RETURN to the main routine.

860 Return.

870 Again, as in line 720, a decision has to be made as to which 'd' value to use for the loop control. This time I decided to plump for DX first if it was zero. (There are no ulterior motives for plumping one way or the other, by the way. It just happened to be a sunny day...)

880. This time, because one of the 'd' values is zero, its value is going to have no part in the control of the loop; X1 = X2 at this point, so it doesn't really matter whether we use X1 or X2 to set the column value. Once set, it won't change during the loop, so it only needs to be set BEFORE the loop, not during it - which will save time - and so C is assigned the rounded integer value of X1 here.

890 All we need to do now, because there is no slope to bother about (the line is going to be either vertical or horizontal if one of 'd' values is zero), is to run the loop from Y1 to Y2 with a step value of SGN(DY).

900 R is assigned the rounded integer value of the loop control value, I.

910 Plot.

920 End of loop.

930 Return.

940 In line 870, if DY was zero a jump would have been made to this subsection. (If DX is zero, that means that there is no change in the column value, only the row, so the line will be vertical; if DY then no row change, only column, so the line will be horizontal). This next subsection is the parallel of lines 880 - 930. This line assigns R the rounded integer value of Y1.

950 The loop from the starting column to the finishing one is set up, with a step value of +1 or -1 depending upon the value of SGN(DX).

960 C is assigned the rounded integer value of the loop control value. If you have managed to follow all this so far, you may be wondering why we need to take rounded integers if the STEP value is plus or minus 1. (Go on, then, wonder!) The reason is that your values for X1, Y1, X2, Y2 could be REAL numbers (explained last issue), i.e., numbers containing fractions. The loop could be running from 17.46778592 to 22.409362781 with a step value of +1, couldn't it ?

970 Plot.

980 End of loop.

990 Return.

There we are. Simple, isn't it ?! Now for some more really fun stuff. During the development of this subroutine, I accidentally came across a routine which gives the most uncanny beetle-like figure on the screen. If you leave it long enough, and if you've had enough to drink, it ends up looking like the Sun with thin wisps of cloud crossing its surface. (Boy,have you got to be imaginative in this game!!) The Beetle is listed below the DRAW subroutine on the listings pages. Don't forget that you need BOTH DRAW and PLOT subroutines to be present to make the thing work.

Next item on the agenda. The Medium Resolution I routine plots like the High Resolution plotter given in the last issue, but places each plotted point on screen as a group of four dots. This means that a) TWO of the digits in the hexadecimal definition string used to define the character in which the point occurs will need to be redefined (one will be two digits away from the other: if the point causes the first hex digit to be redefined, it will also affect the 3rd in the string, and so on) and b) in binary terms, plotting a point is the same as replacing the current PAIR of points (which may be either on or off), so that there are a very limited number of possible patterns of binary to define the points: 0, 3, C, and F, to be exact. They represent the patterns 0000, 0011, 1100, and 1111. If a point is plotted and it occurs in the top left hand corner of a previously unused character, the complete binary pattern looks like this:

```
11000000 = C0 in hex
11000000 = C0
00000000 = 00
00000000 = 00                The first hex string is therefore:
00000000 = 00
00000000 = 00                C0C0000000000000
00000000 = 00
00000000 = 00
```

If the next point is plotted in the bottom right corner, it will affect the pattern thus:

```
11000000 = C0 in hex
11000000 = C0
00000000 = 00
00000000 = 00                The first hex string is therefore:
00000000 = 00
00000000 = 00                C0C0000000000303
00000011 = 03
00000011 = 03
```

As each plotted point affect or involves a pair of binary digits, I thought it might be easier to work out which group of four binary digits were being addressed, and depending upon their present value and the location of the new plotted point, update the hex string WITHOUT the need for a binary reference string. For example, in the first matrix shown, if another point was to be plotted just to the right of the point shown, the two lines affected would change from 1100 to 1111
1100 1111

which in hex terms means that if the present position is defined by a C and the new point is going to lie on the right of the present point (i.e., next door to) then the new position will be defined by an F. There are only two positions in each 4 digit group which can be occupied by the binary pairs, and so the algorithm (I've been using this word quite often without explaining to the uninitiated what it means: essentially it's a rule, or series of rules, of thumb, used to perform a process of some kind. An algorithm can be mathematically based, buy usually it is the "natural" language (i.e., for use, English) original upon which a program is based. For example, "only plot a point if it lies on screen" is part of an algorithm which could find expression in Basic as: IF (R > 24)+(C > 32) THEN jump over the plotting instructions. A more detailed explanation can be given if you want one.) is fairly simple: if the current point is defined by hex C, and if it is in the right pair, the new position is defined by hex 3. And so on for the other possible values 3, C, and F. In full, for left pairs, 0 becomes C, 3 becomes F, C stays as C, and F stays as F. For right pairs, 0 becomes 3, 3 stays as 3, C becomes F, and F stays as F. I have used two reference strings, both called H$, with elements 1 and 2, in other words, H$ is a one-dimensional array. The values for the original possible variations are held in B$, so that having found the hex digit in the

>continued (page retyped 2021)

definition string for the particular character, POS can be used to find its position in B$. Then, having worked out whether the plot is going to affect the left or right binary pair, the same position in either H$(1) or H$(2) will give the relevant new hex digit. This can then replace the original part in the definition string.

Now, to give you something to do other than fall asleep, the version of the medium resolution plotter I've given here is an early prototype. It has since been improved and as a test of your understanding of my explanations (in other words, a test of my ability to explain, or at least, to hold your interest) I want you, dear reader, to see if you can work out where the unnecessary bits are in the subroutine which begins at line 1000. I'll give you a few clues so that newcomers won't feel left too far behind. Firstly, you won't need to construct vastly complex equations to improve things (as I always find myself doing), and secondly, you could delete three lines and still do the same job. I'm sorry that there aren't any wonderful prizes to offer apart from your name in lights (well, in grey letters) on the pages of BB. There's no race, as all entries and clean suggestions will get a lock-in. Send your solutions together with as much documentation as you think you can give, to be at Oxford (address given earlier in the article). Oh, and one further clue: two of the lines you could delete are not REMs! You can of course delete more; as I'm overly fond of saying, the only thing that limits you is your imagination.

All of the example routines which were given last issue could be modified to run with this subroutine (don't forget that the initialization is different); bear in mind that the plotting area, although it is the same size on screen, has only a half of the range of the previous plotters; rows 1 to 96 and columns 1 to 128. This could make the three-dimensional plots look a little crowded.

Because of the task I have set those of you who are interested, I will not be giving a detailed analysis of the subroutine line by line or that would give the whole game away! Those of you who would rather wait for the better version will have to wait at least until the next issue or even longer, depending upon the response. The more or you respond, the sooner I can publish the improved version(s).

Next!

A further development was the Medium Resolution II (I'd hate to break an American tradition), in which plotted points are bigger, better, and faster! The basic outline for this has been given earlier. As each one of those shapes in Figure 1 can be given a unique four digit binary representation (0000 to 1111) and each shape can be pre-defined and assigned to a unique character, I have used a similar system to the one used for the high resolution plotted in that GCHAR is used to find out which character currently occupies the position in which a point is to be plotted; the code of that character is held in a reference string (C$), and its position corresponds to the four digit group of binary numbers held in B$. Having obtained the requisite binary pattern, and modified it (even if the point to be plotted has already been plotted — it's quicker that way), the new binary pattern's position is found in B$ and that position corresponds to the ASCII code of the relevant character held in C$. (I should emphasize here that the example routines given last issue must be modified to run with medium resolution I and II — that means that some of the variable names will have to be changed, due mostly to the fact that they were developed separately from these routines, and also due in part to the fact that I am still developing much of the material which I present here). Please note that because this system uses pre-defined character shapes, the screen and TP dumps will not work, nor will the axes — with or without protection, unless you develop your own, and please send them in if you do — and the three dimensional plots will look less than agreeable. I know this because I have tried running them on a Sinclair, which has the same degree of plotting resolution, and they don't look anything like as good as in high resolution.

The detailed explanation follows:-

100 Clear the screen. This puts character 32, the space over the entire screen, and this is intentional. The space (which doesn't need to be redefined as a blank, 0000000000000000, because it already is so) sets the screen to an unplotted condition.

110 Reversing previous display conditions, I have made this a black on grey, rather than white on black, plot. The screen is therefore set to colour code 15, grey.

120, 130, and 140 hold the DATA which will be used to predefine the necessary shapes. They shouldn't need explanation as the handbook deals adequately with character definition. If you think otherwise, please write in and give us your views and air your problem(s).

150 Set up a loop to READ the items in the DATA list and define the characters. I have decided to take characters with codes 32 to 47 inclusive as my characters, and as 32 is already blank, only 33 to 47 need to be redefined. Note that loops don't have to run from 0 or 1 to another number.

160 This line reads in an item from the DATA list and assigns it to D$. D$ will only hold that item for as long as is necessary to use it to define a given character.

170 Define the character using CALL CHAR. The loop control value, I, holds the ASCII code of the character we want to define with the hex definition string, currently held in D$.

180 End of loop.

190 The sixteen characters used (ASCII 32 to 47) span two colour sets, 1 & 2. This and line 200 define the foreground and background colours for those two sets as black foreground and grey background.

200 See above.

210 This is the binary reference string, which holds the binary equivalent of the shapes given in Figure 1. (Note that this is NOT the binary equivalent of the definition strings used.)

220 This is the ASCII code string, with requisite separators (full stops) so that the ASCII code position in the string corresponds with its binary shape equivalent in B$.

Your own routines (the random plotters and user-directed plotters are probably best for this system) begin from line 230 onwards. (Line 210 in the Medium Resolution I system).

The subroutine:-

1000 Calculates the screen row from the 'virtual' row value.

1010 Does the same for the columns.

1020 Uses GCHAR at those 'real' screen coordinates to contain the ASCII code of the character there, and assigns it to numeric variable H.

1030 Finds the position in C$ of the ASCII code of the character it has just obtained. The first digit of that code is at the same position in C$ as the first digit of the binary representation of the character is in B$.

This segment of B$ is then extracted using SEG$ and assigned to E$.

1040 This line determines which of the 4 bits (binary digits) needs to be changed to a "1". The whys and wherefores of making such calculations will be covered in the Maths sections of future BB articles.

1050 The "1" is inserted in the correct position in the binary sequence, and the result stored back in H$.

1060 That new binary sequence occurs somewhere in the B$ string. POS is used to find the start location of that sequence within B$, and that also is the position within C$ of the relevant ASCII code of the character whose shape has been predefined according to that binary sequence. That ASCII code is extracted, and as it is a string, is operated on by VAL to turn it into a form that the computer will recognise as valid for use in a numeric variable. This is assigned to H.

1070 The character with that code is then placed on the screen, replacing the previous character.

1080 Return.

Before going on to the sections on the Replicator and a short book review, here are a couple of hints which may be of help to some of you. If you have SAVEd some program to CS2, how can you check that it has gone on tape OK? The answer is to key SAVE CS1, having rewound the tape to the beginning of the program and transferred it to CS1 either by changing tape recorders or swapping leads, press ENTER then press C to Check and follow the instructions. Because my machine is an early 99/4 of the NTSC variety, my machine will keep on asking me if I want to check a program SAVEd to CS2, even though the leads to do it aren't there. (Actually, they are: my lead is also an early one, and the earphone lead has simply been cut off. You can still see the bare wires sticking out of the end. One of these days I will get around to reinstating the CS2 lead so that it is the same as the CS1 lead; my old and rather expensive 99 will then have a facility that the latest 4As and some of the PAL 4s lack.)

The second hint refers to aligning the numbers you print out so that all the decimal points lie under one another. This came from Sheridan Williams, who writes for Personal Computer World, in an old copy of PCW (did you know that PCW once published an article on building your own 9900-based system?). The August 1979 issue is the one to look at for the alignment info.

For alignment use the definition: `DEF FMA(X) = -LEN(STR$(INT(X)))-(ABS(X)<1)`
Then, to print out numbers with their decimal points all sitting on a particular point on the screen, use `PRINT TAB(Y-FMA(X)): X`
where X is your number or numeric variable/array, and Y is the TAB position at which the decimal point is to be placed.
To make a right-hand justification, use `DEF FMA(X) = 1 - LEN(STR$(X))` instead, and use the same PRINT statement.

A couple more things. During the creation (!) of this article I have received a couple of listings from members Peter Phillips (Glasgow) and Stephen Shaw (Stockport), which I have included without detailed explanation at the end of the article. Stephen's uses Extended Basic, and if you don't possess the RAM expansion, delete the lines which CALL INIT. I don't have Extended Basic (I was kindly lent a copy by member Steve Fung to test), but Stephen tells me that they are good fun. The listing from Peter Phillips uses the DRAW command (I often try out some of my ideas on members who write to me) and was just a quick sample from him of what he had been doing with DRAW, with the possibility of more to come when he can find time.

retyped 2021: last 3 paras reformatted.

SCIENCE WATCH

In the March 18, 1979 issue (No. 130) of the Telegraph's Sunday Magazine, on page 106, an all-too-short article by Kenneth Gatland appeared, entitled: "See You Later Replicator". I have searched high and low for more information, but this is all I have to date. The article concerned the subject of a USA patent No. 4, 078, 229. The subheading reads: "Show this machine anything you want, and it will make it for you..."

In 1977, a young American inventor, Wyn Kelly Swainson, who worked his ideas out at the Woolwich Polytechnic, showed how the methods of photography and chemistry could be used to make a new kind of automated machine, the "Replicator," which could construct, in three dimensions, any kind of complicated shaped object from engineering descriptions. Already lathes and milling machines are being controlled by instructions fed from computers obeying engineering specifications, but this is something different. For one thing, there are few, if any moving parts. For another, its basic raw material is not a solid but a solution...

Swainson had discovered that there are certain chemicals, which, when in solution, can have their structure changed by contact with light. This in itself was nothing dramatic; after all, many substances exist which change on exposure to light — the silver in photographic emulsions, for example. What was dramatic was the way in which Swainson set about exposing these chemicals to light. He had discovered that there were certain plastics which could be dissolved, and made to "precipitate" when two light beams were passed through the solution. The plastic would only appear in a lump at the point where the two beams of light crossed; the beams were of differing wavelengths. He used the very fine beams of light which are produced by modern lasers, and controlled them with a computer. Because no great mechanical force is involved (unlike with tools paring away metal or plastic), the Replicator can be built to be light and cheap. Swainson also discovered that, with the correct combinations of lasers of different frequency, he could get certain metals to come out of solution, always at the point where the beams crossed.

If you have one of these little marvels in your home, the idea is that all you have to do is supply the power source (and with the new high power sources, and smaller lasers, that's not going to be a big deal) and the solutions to keep topping up the tank in which the work is done. To get a spare part (of a convenient size, of course: I don't think the domestic one will make an engine block for you!) for something, all you do is ring up your local dealer who sends the relevant data down the line to your Replicator, which make a one-off of the thing you require. And if you can do it for plastic and metal, you could conceivably do it for ceramics, as they are only mineral salts anyway...

The article goes on to speculate further about furniture and clothing, and because the system uses solutions, you can recycle any damaged or worn items by dissolving them and sticking them back in the tank with some fresh solution. For industrial use, put a hundred of these on a larger, and you can make a hundred items simultaneously, engine blocks and all. What are currently waste products like cans and plastic bottles, would become raw material for the Replicator.

When are we likely to get one for the home? It seems that first the industrial users have to have their look-in, and the first machines are supposed to be designed for use in factories and foundries. The domestic versions could follow in a few years. However, a few years has already elapsed, and I've seen no evidence of the machine appearing in any of the journals I read, nor has it appeared, to the best of my knowledge, on Horizon or Tomorrow's World. I can foresee problems in things like quality control (how can you be sure that you've mixed the chemicals right?) and sterility (how do you keep pathological organisms out of items which might be used for food?) not least of which would be chemical contamination, as these

chemicals are likely to be, if not highly toxic, then at least not very good for you if you happened to swallow them. Any item coming out of such a system would probably need to be thoroughly washed before use. Anyone who has any doubts about the ability of today's computers to handle three-dimensional processing obviously hasn't seen the Horizon program on computer graphics. One of the companies who were involved in 1979 were Batelle Columbus Laboratory, a household name if you have a household with a Xerox copier in it, because that's what they were responsible for, among other things, and they are (in 1979 anyway) heavily involved in new technology.

If anyone has anything further they'd like to add, please write in to me at the address given earlier.

BOOK REVIEW

It's just coincidence, but in the month I decide to review Everything You Want To Know About Personal Computers by Peter Rodwell, editor of Personal Computer World, PCW themselves reviewed it as well. I can refer you to the April issue of PCW is you want to see a different viewpoint. As far as I'm concerned, it is the book that I would have written had I got off my backside and got around to doing something about it. It's not just that the style is the one I would have chosen. (Although I wouldn't have countenanced the typographical errors which really only serve to further confuse those looking for a little enlightenment.) Many of the ideas which I have on the "shape of things to come", and the uses to which I at least would put some of the new pieces of technology on the market, are identical to those presented by Peter Rodwell, especially in relation to the use of video discs — see the book review in the last issue of Tidings. (Barry Sherman of ASTMS is another individual whose publicly-declared ideas on the future agree closely with mine — if any of you saw Everyman the other evening, you will have seen what I mean). Peter Rodwell's book covers most basic topics in computing (at least, at our end) but not in the kind of verbose torrent that my articles seem to generate. Obviously I am biased towards his ideas, not least because it is not new ground to me, but it should help ease some of you into the amateur micro world. And at £1.50 it won't break the bank.

Unlike the second book I was going to review, and which is currently defying some of my attempts to understand it. At £10.45 ($14.95 in the States), 16 Bit Microprocessors is not exactly cheap, but it does have a chapter with the layman in mind about the 9900 family of chips. I've understood some of the machine code instructions which are described, but until I can get some "hands-on" experience, (hopefully with the new mini-assembler when it appears AND if it's not vastly overpriced, as usual) I can't really say whether this book is a good idea for those of you longing to take to the air with fast Space Invaders. Watch this space...

That about wraps it up for this issue. I just hope I can get it all to Paul Dicks on time...

Good programming,

Pete Brooks

```
10 REM  MODIFIED PLOTTING SUBROU
TINE & IMPLEMENTATION OF THE "DR
AW" COMMAND
20 REM  PETE BROOKS 1982..TO USE
 'DRAW',SET Y1,X1;Y2,X2 TO START
 & FINISH LOCATIONS ON SCREEN &
'GOSUB 690'
30 REM  RESERVED VARIABLES:B,C,D
X,DY,H,I,L,P,R,S,X,X1,X2,Y,Y1,Y2
,B$,C$(),H$,I$,P$,Z$
40 REM  EITHER INSERT YOURROUTIN
ES BETWEEN 200 & 680 USING GOSUB
 690 FOR DRAW,& GOSUB 1000 FOR P
LOT
50 REM  OR MAKE LINE 200 INTO 'G
OTO 1230',MAKE 1230 INTO 'REM',
AND THEN RESEQUENCE.YOUR ROUTINE
S CAN NOW BE
60 REM  PLACED FROM WHAT WAS 123
0,ONWARDS,WITH RELEVANT CHANGES
TO THE GOSUBS.
70 REM  'DRAW' WORKS WITH ALL 3
DEGREES OF RESOLUTION
```

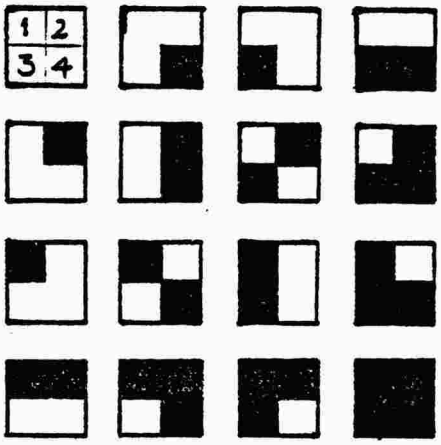


Fig.1

```
100 CALL SCREEN(2)
110 S=31
120 CALL HCHAR(1,1,S,768)
130 DIM C$(128)
140 B$="0000.0001.0010.0011.0100
.0101.0110.0111.1000.1001.1010.1
011.1100.1101.1110.1111"
150 H$="0....1....2....3....4...
.5....6....7....8....9....A....B
....C....D....E....F"
160 Z$="00000000000000000"
170 FOR I=1 TO 16
180 CALL COLOR(I,16,2)
190 NEXT I
690 DX=X2-X1
700 DY=Y2-Y1
710 IF (DX=0)+(DY=0)THEN 850
720 IF ABS(DX)>ABS(DY)THEN 790
730 FOR L=Y1 TO Y2 STEP SGN(DY)
740 R=INT(.5+L)
750 C=INT(.5+X1+DX/DY*(L-Y1))
760 GOSUB 1000
770 NEXT L
780 RETURN
790 FOR L=X1 TO X2 STEP SGN(DX)
800 C=INT(.5+L)
810 R=INT(.5+Y1+DY/DX*(L-X1))
820 GOSUB 1000
830 NEXT L
840 RETURN
850 IF (DX=0)-(DY=0)THEN 870
860 RETURN
870 IF DY=0 THEN 940
880 C=INT(.5+X1)
890 FOR L=Y1 TO Y2 STEP SGN(DY)
900 R=INT(.5+L)
910 GOSUB 1000
920 NEXT L
930 RETURN
940 R=INT(.5+Y1)
950 FOR L=X1 TO X2 STEP SGN(DX)
960 C=INT(.5+L)
970 GOSUB 1000
980 NEXT L
990 RETURN
1000 Y=INT(R/8+.875)
1010 X=INT(C/8+.875)
1020 CALL GCHAR(Y,X,H)
1030 REM  IF POS(P$,CHR$(H),1) T
HEN 1220
1040 IF H>31 THEN 1120
1050 IF S=159 THEN 1220
1060 S=S+1
1070 REM  IF POS(P$,CHR$(S),1) T
HEN 1060
1080 C$(S-31)=Z$
1090 CALL CHAR(S,Z$)
1100 CALL HCHAR(Y,X,S)
1110 H=S
1120 H=H-31
1130 B=C-X*8+8
1140 P=2*R-16*Y+16+(B<5)
1150 IF B<5 THEN 1170
1160 B=B-4
1170 I$=SEG$(B$,POS(H$,SEG$(C$(H
),P,1),1),4)
1180 I$=SEG$(I$,1,B-1)&"1"&SEG$(
I$,B+1,4-B)
1190 I=POS(B$,I$,1)
1200 C$(H)=SEG$(C$(H),1,P-1)&SEG
$(H$,I,1)&SEG$(C$(H),P+1,16-P)
1210 CALL CHAR(H+31,C$(H))
1220 RETURN
```

2021 reminder- read rem line 40!

```
10 REM  'DRAW' SUBROUTINE
20 REM  RESERVED VARIABLES:C.L.R
.DX.DY.X1.X2.Y1.Y2

690 DX=X2-X1
700 DY=Y2-Y1
710 IF (DX=0)+(DY=0)THEN 850
720 IF ABS(DX)>ABS(DY)THEN 790
730 FOR L=Y1 TO Y2 STEP SGN(DY)
740 R=INT(.5+L)
750 C=INT(.5+X1+DX/DY*(L-Y1))
760 GOSUB 1000
770 NEXT L
780 RETURN
790 FOR L=X1 TO X2 STEP SGN(DX)
800 C=INT(.5+L)
810 R=INT(.5+Y1+DY/DX*(L-X1))
820 GOSUB 1000
830 NEXT L
840 RETURN
850 IF (DX=0)-(DY=0)THEN 870
860 RETURN
870 IF DY=0 THEN 940
880 C=INT(.5+X1)
890 FOR L=Y1 TO Y2 STEP SGN(DY)
900 R=INT(.5+L)
910 GOSUB 1000
920 NEXT L
930 RETURN
940 R=INT(.5+Y1)
950 FOR L=X1 TO X2 STEP SGN(DX)
960 C=INT(.5+L)
970 GOSUB 1000
980 NEXT L
990 RETURN
```

```
10 REM  'BEETLE' USING 'DRAW' SU
BROUTINE

200 FOR Z=90 TO 160 STEP 2
210 Y1=96+SIN(Z/9)*10
220 X1=96+COS(Z/9)*10
230 Y2=96+SIN(Z/3)*20
240 X2=96+COS(Z/9)*20
250 GOSUB 690
260 NEXT Z
270 GOTO 270
```

```
10 REM  MEDIUM RESOLUTION I

20 REM   RESERVED VARIABLES:B,C,
H,P,R,S,X,Y,B$,C$(),H$(),I$,J$,P
$,Z$

100 CALL SCREEN(2)
110 S=31
120 CALL HCHAR(1,1,S,768)
130 DIM C$(128)
140 B$="03CF"
150 H$(1)="CFCF"
160 H$(2)="33FF"
170 Z$="00000000000000000"
180 FOR I=1 TO 16
190 CALL COLOR(I,16,2)
200 NEXT I

1000 Y=INT(R/4+.75)
1010 X=INT(C/4+.75)
1020 CALL GCHAR(Y,X,H)
1030 REM  IF POS(P$,CHR$(H),1) T
HEN 1230
1040 IF H>31 THEN 1120
1050 IF S=159 THEN 1230
1060 S=S+1
1070 REM  IF POS(P$,CHR$(S),1) T
HEN 1060
1080 C$(S-31)=Z$
1090 CALL CHAR(S,Z$)
1100 CALL HCHAR(Y,X,S)
1110 H=S
1120 H=H-31
1130 B=C-X*4+4
1140 P=4*R-16*Y+14+(B<3)
1150 IF B<3 THEN 1170
1160 B=B-2
1170 I$=SEG$(C$(H),P,1)
1180 J$=SEG$(C$(H),P+2,1)
1190 I$=SEG$(H$(B),POS(B$,I$,1),
1)
1200 J$=SEG$(H$(B),POS(B$,J$,1),
1)
1210 C$(H)=SEG$(C$(H),1,P-1)&I$&
SEG$(C$(H),P+1,1)&J$&SEG$(C$(H),
P+3,14-P)
1220 CALL CHAR(H+31,C$(H))
1230 RETURN
```

```
10 REM  MEDIUM RESOLUTION II
20 REM  RESERVED VARIABLES:C,H,P
.X,Y,B$,C$,H$

100 CALL CLEAR
110 CALL SCREEN(15)
120 DATA 00000000F0F0F0F,000000
00F0F0F0F,00000000FFFFFFFF,0F0F0
F0F,0F0F0F0F0F0F0F0F,0F0F0F0FF0F
0F0F
130 DATA 0F0F0F0FFFFFFFFF,F0F0F0
F,F0F0F0F00F0F0F0F,F0F0F0F0F0F0F
0F,F0F0F0F0FFFFFFFF,FFFFFFFF,FFF
FFFFFF0F0F0F0F
140 DATA FFFFFFFFF0F0F0F,FFFFFFF
FFFFFFFFF
150 FOR I=33 TO 47
160 READ D$
170 CALL CHAR(I,D$)
180 NEXT I
190 CALL COLOR(1,2,15)
200 CALL COLOR(2,2,15)
210 B$="0000.0001.0010.0011.0100
.0101.0110.0111.1000.1001.1010.1
011.1100.1101.1110.1111"
220 C$="32...33...34...35...36..
.37...38...39...40...41...42...4
3...44...45...46...47"

1000 Y=INT(R/2+.5)
1010 X=INT(C/2+.5)
1020 CALL GCHAR(Y,X,H)
1030 H$=SEG$(B$,POS(C$,STR$(H),1
),4)
1040 P=C+2*R-2*X-4*Y+4
1050 H$=SEG$(H$,1,P-1)&"1"&SEG$(
H$,P+1,4-P)
1060 H=VAL(SEG$(C$,POS(B$,H$,1),
2))
1070 CALL HCHAR(Y,X,H)
1080 RETURN
```

```
30 REM FREEFORM ART
   BY STEPHEN SHAW 1982          S
UBPROGRAMS BY:
40 REM PETER BROOKS  AND
50 REM JEREMY RUSTON
60 CALL CLEAR :: PRINT "FREE FOR
M ART":"1982":"STEPHEN SHAW":"PE
TER BROOKS":"JEREMY RUSTON": : :
70 PRINT "RANDOM DESIGN":"CURSOR
 WILL APPEAR":"AT TOP WHEN NO MO
RE CHARS &":"AT BOTTOM WHEN DESIGN
 ENDED": :"PRESS ENTER FOR ANOTH
ER"
72 PRINT "DESIGN": :
80 ACCEPT AT(24,20):Q$ :: CALL C
LEAR
100 REM RESERVED:B1 C  H1 I1 P1
R S1 X1 Y1 B$ C$() H$ I$ P$ Z$
     R<192 C<256          USE GOSUB
 29700    ***PLOT*
110 CALL SCREEN(16):: S1=31 :: C
ALL HCHAR(1,1,S1,768):: DIM C$(1
20):: B$="0000.0001.0010.0011.01
00.0101.0110.0111.1000.1001.1010
.1011"
120 B$=B$&".1100.1101.1110.1111"
 :: H$="0123456789ABCDEF" :: Z$=
RPT$("0",16)
130 RANDOMIZE :: CALL HCHAR(1,1,
31,768):: S=31 ! RESTART HERE
140 XX=INT(RND*110+40):: YY=INT(
RND*110+40):: LL=INT(RND*110+40)
:: MM=INT(RND*110+40):: UU=(10-R
ND*20)
150 VV=(10-RND*20):: PP=(10-RND*
20):: QQ=(10-RND*20):: FOR K=1 T
O 30
160 X2=XX :: Y2=YY :: X3=LL :: Y
3=MM :: GOSUB 240
170 IF XX+UU>150 OR XX+UU<40 THE
N UU=-UU
180 IF YY+VV>150 OR YY+VV<41 THE
N VV=-VV
190 IF LL+PP>150 OR LL+PP<39 THE
N PP=-PP
200 IF MM+QQ>148 OR MM+QQ<38 THE
N QQ=-QQ
210 XX=XX+UU :: YY=YY+VV :: LL=L
L+PP :: MM=MM+QQ :: NEXT K
220 ACCEPT AT(23,4):KEY$
230 RANDOMIZE :: GOTO 130
240 DX=X3-X2 :: DY=Y3-Y2 :: IF (
DX=0)+(DY=0)THEN 280
250 IF ABS(DX)>ABS(DY)THEN 270
260 FOR LCV=Y2 TO Y3 STEP SGN(DY
):: R=INT(.5+LCV):: C=INT(.5+X2+
DX/DY*(LCV-Y2)):: GOSUB 340 :: N
EXT LCV :: RETURN
270 FOR LCV=X2 TO X3 STEP SGN(DX
):: C=INT(.5+LCV):: R=INT(.5+Y2+
DY/DX*(LCV-X2)):: GOSUB 340 :: N
EXT LCV :: RETURN
280 IF (DX=0)-(DY=0)THEN 300
290 RETURN
300 IF DY=0 THEN 320
310 C=INT(.5+X2):: FOR LCV=Y2 TO
 Y3 STEP SGN(DY):: R=INT(.5+LCV)
:: GOSUB 340 :: NEXT LCV :: RETU
RN
320 R=INT(.5+Y2):: FOR LCV=X2 TO
 X3 STEP SGN(DX):: C=INT(.5+LCV)
:: GOSUB 340 :: NEXT LCV :: RETU
RN
330 STOP
340 REM PLOT SUBROUTINE
350 Y1=MIN(INT(R/8+.875),24):: X
1=MIN(INT(C/8+.875),32):: CALL G
CHAR(Y1,X1,H1):: IF H1>31 THEN 3
80
360 IF S1>=143 THEN CALL SOUND(9
00,440,0):: ACCEPT AT(1,1)SIZE(-
2)BEEP:ZED$ :: GOTO 130
370 S1=S1+1 :: C$(S1-31)=Z$ :: C
ALL CHAR(S1,Z$):: CALL HCHAR(Y1,
X1,S1):: H1=S1
380 H1=H1-31 :: B1=C-X1*8+8 :: P
1=2*R-16*Y1+16+(B1<5):: IF B1<5
THEN 400
390 B1=B1-4
400 I$=SEG$(B$,POS(H$,SEG$(C$(H1
),P1,1),1)*5-4,4):: I$=SEG$(I$,1
,B1-1)&"1"&SEG$(I$,B1+1,4-B1)::
I1=POS(B$,I$,1)/5+.8
410 C$(H1)=SEG$(C$(H1),1,P1-1)&S
EG$(H$,I1,1)&SEG$(C$(H1),P1+1,16
-P1):: CALL CHAR(H1+31,C$(H1))
420 RETURN
```

```
40 CALL CLEAR :: PRINT "ETCH A S
KETCH TYPE HIGH":"RESOLUTION DRA
WING PROG"
50 PRINT " ":"STEPHEN SHAW 1982"
:"PETER BROOKS":: :"USE THE 8 ARR
OW KEYS TO MOVETHE PEN(LIGHT RED
 DOT)":"PRESS KEY 1 TO LEAVE A T
RACE"
60 PRINT "& KEY 2 TO SWITCH TRAC
E OFF":: :"PRESS ENTER TO CONTI
NUE" :: INPUT A$ :: CALL CLEAR
100 REM KEYBOARD PLOTTER
110 CALL COLOR(0,11,11)
120 CALL INIT :: CALL LOAD(-3187
8,2)
130 REM RESERVED:B1 C  H1 I1 P1
R S1 X1 Y1 B$ C$() H$ I$ P$ Z$
     R<192 C<256          USE GOSUB
 29700     ***PLOT*
140 CALL SCREEN(2):: S1=32 :: CA
LL HCHAR(1,1,170,768):: DIM C$(1
20):: B$="0000.0001.0010.0011.01
00.0101.0110.0111.1000.1001.1010
.1011"
150 CALL CHAR(32,"404")
160 CALL HCHAR(1,1,31,128):: CAL
L HCHAR(21,1,31,128):: CALL VCHA
R(1,1,31,140):: CALL VCHAR(1,28,
31,140)
170 PR=0
180 B$=B$&".1100.1101.1110.1111"
 :: H$="0123456789ABCDEF" :: Z$=
RPT$("0",16):: FOR IQ=1 TO 14 ::
 CALL COLOR(IQ,16,2):: NEXT IQ
190 R=96 :: C=128
200 CALL SPRITE(#1,32,10,96,128)
210 CALL KEY(1,K,T):: IF T=0 THE
N 210 ELSE R=R+(K=4)+(K=5)+(K=6)
-(K+1=1)-(K=14)-(K=15):: C=C+(K=
2)+(K=4)+(K=15)-(K=3)-(K=6)-(K=1
4)
220 IF K=19 THEN PR=1 ELSE IF K=
7 THEN PR=0
230 CALL LOCATE(#1,R,C)
240 IF PR=0 THEN 210
250 GOSUB 260 :: GOTO 210
260 REM PLOT SUBROUTINE
270 Y1=MIN(INT(R/8+.875),24):: X
1=MIN(INT(C/8+.875),32):: CALL G
CHAR(Y1,X1,H1):: IF H1<144 THEN
300 ELSE IF H1>150 THEN H1=32
280 IF S1>=143 THEN CALL SOUND(9
00,440,0):: ACCEPT AT(1,1)BEEP:Z
ED$ :: GOTO 190
290 S1=S1+1 :: C$(S1-31)=Z$ :: C
ALL CHAR(S1,Z$):: CALL HCHAR(Y1,
X1,S1):: H1=31
300 IF H1=31 THEN 340 ELSE H1=H1
-31 :: B1=C-X1*8+8 :: P1=2*R-16*
Y1+16+(B1<5):: IF B1<5 THEN 320
310 B1=B1-4
320 I$=SEG$(B$,POS(H$,SEG$(C$(H1
),P1,1),1)*5-4,4):: I$=SEG$(I$,1
,B1-1)&"1"&SEG$(I$,B1+1,4-B1)::
I1=POS(B$,I$,1)/5+.8
330 C$(H1)=SEG$(C$(H1),1,P1-1)&S
EG$(H$,I1,1)&SEG$(C$(H1),P1+1,16
-P1):: CALL CHAR(H1+31,C$(H1))
340 RETURN
```

```
10 CALL CLEAR :: PRINT "TRIANGLE
S":: :"1982":: :"STEPHEN SHAW":"PE
TER BROOKS":"JEREMY RUSTON":: :"R
ANDOMIZED DESIGNS":: :"CURSOR WIL
L APPEAR AT"
20 PRINT "TOP WHEN NO MORE CHARS
 &":"AT BOTTOM WHEN PICTURE":"IS
 FINISHED":: :"PRESS ENTER FOR AN
OTHER":"DESIGN":: ::
50 RANDOMIZE :: ACCEPT AT(24,24)
BEEP:G$
100 REM RESERVED:B1 C  H1 I1 P1
R S1  X1 Y1 B$ C$() H$ I$ P$ Z$
      R<192 C<256          USE GOSUB
 29700      ***PLOT*
110 CALL SCREEN(2):: S1=31 :: CA
LL HCHAR(1,1,31,768):: DIM C$(12
0):: B$="0000.0001.0010.0011.010
0.0101.0110.0111.1000.1001.1010.
1011"
120 B$=B$&".1100.1101.1110.1111"
 :: H$="0123456789ABCDEF" :: Z$=
RPT$("0",16):: FOR I9=1 TO 14 ::
 CALL COLOR(I9,16,2):: NEXT I9
130 S1=31 :: RANDOMIZE :: AA=RND
*110+40 :: BB=RND*110+40 :: CC=R
ND*110+40 :: DD=RND*110+40 :: EE
=RND*110+40 :: FF=RND*110+40
140 UU=28*RND-14 :: VV=28*RND-14
 :: WW=28*RND-14 :: XX=28*RND-14
 :: YY=28*RND-14 :: ZZ=28*RND-14
150 FOR KK=1 TO 10
160 X2=AA :: Y2=BB :: X3=CC :: Y
3=DD :: GOSUB 270
170 X2=X3 :: Y2=Y3 :: X3=EE :: Y
3=FF :: GOSUB 270
180 X2=X3 :: Y2=Y3 :: X3=AA :: Y
3=BB :: GOSUB 270
190 IF AA+UU>150 OR AA+UU<38 THE
N UU=-UU
200 IF BB+VV>151 OR BB+VV<38 THE
N VV=-VV
210 IF CC+WW>151 OR CC+WW<38 THE
N WW=-WW
220 IF DD+XX>151 OR DD+XX<38 THE
N XX=-XX
230 IF EE+YY>151 OR EE+XX<38 THE
N YY=-YY
240 IF FF+ZZ>151 OR FF+ZZ<39 THE
N ZZ=-ZZ
250 AA=AA+UU :: BB=BB+VV :: CC=C
C+WW :: DD=DD+XX :: EE=EE+YY ::
FF=FF+ZZ
260 NEXT KK :: ACCEPT AT(24,20):
AAA$ :: CALL HCHAR(1,1,31,768)::
 GOTO 130
270 DX=X3-X2 :: DY=Y3-Y2 :: IF (
DX=0)+(DY=0)THEN 310
280 IF ABS(DX)>ABS(DY)THEN 300
290 FOR LCV=Y2 TO Y3 STEP SGN(DY
):: R=INT(.5+LCV):: C=INT(.5+X2+
DX/DY*(LCV-Y2)):: GOSUB 370 :: N
EXT LCV :: RETURN
300 FOR LCV=X2 TO X3 STEP SGN(DX
):: C=INT(.5+LCV):: R=INT(.5+Y2+
DY/DX*(LCV-X2)):: GOSUB 370 :: N
EXT LCV :: RETURN
310 IF (DX=0)-(DY=0)THEN 330
320 RETURN
330 IF DY=0 THEN 350
340 C=INT(.5+X2):: FOR LCV=Y2 TO
 Y3 STEP SGN(DY):: R=INT(.5+LCV)
:: GOSUB 370 :: NEXT LCV :: RETU
RN
350 R=INT(.5+Y2):: FOR LCV=X2 TO
 X3 STEP SGN(DX):: C=INT(.5+LCV)
:: GOSUB 370 :: NEXT LCV :: RETU
RN
360 STOP
370 REM PLOT SUBROUTINE
380 Y1=MIN(INT(R/8+.875),24):: X
1=MIN(INT(C/8+.875),32):: CALL G
CHAR(Y1,X1,H1):: IF H1>31 THEN 4
10
390 IF S1>=143 THEN CALL SOUND(9
00,440,0):: ACCEPT AT(1,1)BEEP S
IZE(-2):ZED$ :: CALL HCHAR(1,1,3
1,768):: GOTO 130
400 S1=S1+1 :: C$(S1-31)=Z$ :: C
ALL CHAR(S1,Z$):: CALL HCHAR(Y1,
X1,S1):: H1=S1
410 H1=H1-31 :: B1=C-X1*8+8 :: P
1=2*R-16*Y1+16+(B1<5):: IF B1<5
THEN 430
420 B1=B1-4
430 I$=SEG$(B$,POS(H$,SEG$(C$(H1
),P1,1),1)*5-4,4):: I$=SEG$(I$,1
,B1-1)&"1"&SEG$(I$,B1+1,4-B1)::
I1=POS(B$,I$,1)/5+.8
440 C$(H1)=SEG$(C$(H1),1,P1-1)&S
EG$(H$,I1,1)&SEG$(C$(H1),P1+1,16
-P1):: CALL CHAR(H1+31,C$(H1))
450 RETURN
```

From Peter F. Phillips of Glasgow, using the high resolution plotter and the DRAW subroutine: (note that Peter has used the letter O as a variable name; to distinguish it on this typeface I will use 'o' in its place)

```
200  FOR o = 1 TO 5
210  READ X1, Y1, X2, Y2
220  NN = 2
230  GOSUB 400
240  NEXT o
250  ZZ = ZZ + 1
260  IF ZZ = 2 THEN 2000
270  RESTORE
280  SHX = 30
290  SHY = 20
300  FOR o = 1 TO 5
310  READ X1, Y1, NII, NUSH
320  X2 = X1 + SHX
330  Y2 = Y1 + SHY
340  GOSUB 690
350  NEXT o
360  RESTORE
370  GOTO 200
380  DATA  60,60,60,120,60,120,120,120,
     120,120,120,60,120,60,80,40,80,40,
     60,60
400  X1 = X1 + SHX
410  X2 = X2 + SHX
420  Y1 = Y1 + SHY
430  Y2 = Y2 + SHY
2000  GOTO 2000
```

Peter's current project is to produce a structural frame analysis program using matrix inversion. It is apparently now at the stage where it will analyse continuous beams. If any of you are engineers, this could be of use to you...

2021 note- not much ink on this one. PB has replaced the original variable O with an o, so where you see for example NEXT o, make it easy and replace the lower case o with something like N.

Random dots

EXBAS

It's arrived at last!! "WHAT??" you might be excused for saying. Well, dedicated readers of this column (are there any?) will know that I am referring to the elusive ExBas (Extended Basic) Module. The previous non-availability of which I have oft-times bemoaned at great length (maybe someone at TI reads TIdings, and took pity on me, or has the country suddenly been flooded with the said module Anyway, I am now having a lot of fun sheaving thro' no less than 224 pages of Handbook, plus 17 pages of addenda & amendments. Let's hope they've got it right at last! More of ExBas in the next Random Dots.

FAME AT LAST

After being totally (well almost) ignored by the 'popular' computer press it was quite a surprise to see the March edition of PCW featuring the 99.4(a) in a big way (including a quite reasonable benchtest). Incidentally, did you see the rubbish written by TIM (Mr-ZX81) HARTNELL in "Your Computer". I wrote to the Editor of YC pointing out some of the glaring errors in the Hartnell's 'review' of the 99.4, but so far he hasn't published the letter (after writing to me assuring me that he would!! along with Hartnell's reply)

4 vs. 4a

Yes there are differences between the 99.4 and the 4a. And not just the keyboard arrangement, either The differences mean that anyone writing 'portable' programs must be very careful that these facts are understood. The 4a has several extra characters available From the keyboard, quite apart from the 'lower case' (actually SMALL CAPS). For example the bracket pair (square), the reverse slant, tilde, brace pair etc. Also the 4a has 31 so-called 'control' characters (for telecommunications? use). Other points to watch (according to 99'er Mag) are not to use SHIFT or ENTER Keys as Fire buttons in games, or the X and M keys as direction indicators used with CALL KEY Subroutine. So be warned if you are writing programs for publication, or even for the Library. It is interesting to note that the 99.4 is actually selling For MORE than the 4a in USA!!
I have had a few bursts on a 4a in a local dealers and I am not greatly impressed. As a one-time Typewriting Instructor I find the so-called Mickey-Mouse keyboard of the 99.4 VERY EASY to use, when you are accustomed to the non-typewriter function keys. While on the subject of keyboards, I personally think that a lot of swaddle is written about 'non-standard' arrangements. I have yet to see, either in the Flesh or on TV, a single computer operator (apart From Word Processor operators - who are typists after all) able to use more than about 2.5 Fingers on the keyboard, so what price 'IBM Standard' (what ever that is supposed to mean!) keyboards.

CRASHES & GLITCHES (cont)

To add a bit more info on Crashes to Stephen Shaw's handy article on the subject may I add the following: If your program uses the Speech Synth then check out that the Synth is working BEFORE you load the prog. (by either using the Speech Editor in immediate mode or keying in a one-line CALL SAY routine). It's a bit of a letdown to load a long prog. from tape only to find that the system goes high on the first speech line. I have also noticed a curious Crash which will not respond to the normal power-down-switch-on-again sequence. When the Thermal Printer is attached and this (rare) condition occurs then one way to handle it is to switch-off, remove the TP power-up (as far as the Title Page & Menu), then switch-off again and finally re-connect the TP. I have found this to work every time.

PAGES & PAGES

Newcomers may have noticed that the layout of 'Random Dots' is different to the rest of TIdings. What I mean is quite apart from the quaint 'typeface' produced by the Thermal Printer combined with my re-designed character set. It's the fact that I have contrived to print in COLUMNS (like a 'real' magazine - if somewhat shorter). The theory is that short lines are more 'readable' and cut out the tendency to have to trace the end of one line to the beginning of the next with a finger. To partially alleviate this it has become the practice to lay out typewritten A4 width text with double-line spacing, but it is not quite successful and it can be quite tiring on the eyes to read more than a couple of pages of full-width text. Printing in columns is something that Word Processors do admirably, but producing columns on an ordinary typewriter is irksome, to say the least. What do you think??

Here's another puzzle:

WORD SEARCH

SUBJECT: COMPUTING

AT LEAST 20 WORDS

O Q G W V F K O I R M I E A
Y E D I G I T A L O P U K N
D E C Y V R Q B D U T E A A
R A F A A U A R K M E P Y L
K N I F F R M M O L P E V O
S I X S Y R R R W L I E C G
I B R I E I E A E S D H T F
D B G E C Y T T O I I P Q X
Y L W X X X W F N P E G C E
P E Z I B I T H P I L P P H
P K Y P V W X J E I R R C G
O R E J A D E X T E O V V G
L A K R M T M C X M L B R Z
F T E R Y G H C O C Z S B Z
K E Y B O A R D H B B K V B

Mike O'Regan

TI NEWS

I understand that Editor/Assembler is due to be released in the next couple of weeks and Pascal about four weeks later. If anyone is interested in the Editor/Assembler and requires information on the subject please feel free to get in touch with me.

I have very little to say in this section because I am hoping that TI will take the opportunity to include their answers to your comments.

LETTERS

I would like to see ways of converting other Basic dialects to TI Basic.

B.G.Smith, 46 Evington Parks Road, Leicester, LE2 1PR.

In particular, I am interested to know of anyone who has written successful statistics programs or graph plotting.

Derrick Setchell, 78 Lyonsdown Road, New Barnet, Herts, EN5 1JL.

We welcome your members to write articles to our magazine, reasonable money will be paid after publication.

T. M. Chan, Modern Electronics, PO Box 2042, Kowloon, Hong Kong.

If I can assist you in any way I shall be glad to do so. I co-ord--inate observations and analyses of variable stars for the British Astronomical Association.

J E Isles, c/o Dept of Transport, Romney House, Marsham Street, SW1.

Please advise where I can obtain Extended Basic as I have great difficulty in buying it.

M J Scales, 22 Alric Ave, New Malden, Surrey, KT3 4JN.

Due to my occupation (sound recordist ITV) I visit the States regularly or one of our crews may do. If you require any TI books etc in the States don't hesitate to contact me.

D. Atkinson, 1 Spring Gardens, Burley-in-Wharfedale, Nr Ilkley, W. Yorks.

It took me some considerable time before I decided to buy my TI 99/4A. I believed, and still believe, it is one of the best computers on the market from the point of view of value for money. I would have thought that a magazine like your own should have been backed by TI financially so that a better format could be produced and professionals empl-oyed to make the articles more easily understood by beginners. I think you have done a marvellous job in getting TIHOME off the ground.

I H Moore,34 Bucknalls Lane, Garston, Watford, Herts, WD2 7NQ.

I wonder if TI have given any reason for the very high prices for software e.g. Editor/Assembler (when available) £115, TI Invaders £40 the prices seem crazy when compared to compatible machines e.g. VIC 20.

K A Wilson, 2 Brickenhole Lane, Walkeringham, Doncaster, S Yorks, DN10 4HX

As an absolute novice to computing my first purchase was a 99/4. Returned the next day due to a faulty keyboard. Replacement OK - except Cassette interface U/S. The retailer - Taylor Wilson - is just about as u/S as the interface.

M J Hartshorne, Address no longer available.

RAMBLES....

march 82

Have you seen the Personal Computer World review of the 99/4A ? Apart from repeating the error of 'no peeks & pokes' (despite listing peek as a keyword) it seemed a reasonable review.
But what did it mean by suggesting TI Basic worked better with no module in ??? It seems no faster on my model- any members have any supporting comments??
The review of games modules was very useful. The issue was March 82 if you need a back issue.
I saw some stickers on their power supply and disk drive - lacking from mine - what are they warning about please??

I have mentioned Creative Computing from time to time- the current issue has just appeared in all our newsagents at £2.05. This is quite a lot more than a sub (surface mail,3 years,£45) but ideal if you just want an odd copy -or to see what the mag is like.

For some months I have been believing my computer incapable of tones below 110 Hz. Silly me. It can go lower-

CALL SOUND(time, 22000,30,22000,30, FREQ , 30, -4, 0)

will produce a sound about 3½ octaves below FREQ, and a tune can be played. With FREQ below 500 the tones are not too much use, but we are extending the range of the computer by 1 to 2 octaves! The difficult thing is to relate these tones to call sound(time,freq,vol).
Anyone out there with a good ear for pitch who could evaluate the relationship to the 3rd decimal place???
(if you use -8 instead of -4,you can still play a tune, but the sound is weird).
You can also use the above line to produce a different 'voice' when playing notes above 110 Hz.

What do you use YOUR Home Computer for? I use mine quite heavily for domestic accounts, evaluating investments- and of course playing games, music , and even a diary system!

Having seen SELFRIDGES listed as a 99/4A dealer, I was interested in seeing their presentation! On a visit to London this month I dropped in- to find a converted 99/4 sitting all on its own with the demonstration module plugged in (yech- what a terrible demonstration! - no peripherals,no sales material, no staff...

Have you visiteed the Science Museum recently- there's a 99/4 casing there (NTSC version)! The demo computers-actually working- are several PETS, an ITT2020/Apple, and an Acorn. The 'hands-on' terminal is linked to a GEC Mainframe - running 'GUESS'. They have the first TI transistors and mention TI as starting DIL ICs. Be nice if they had a working hands-on 99/4A - with all the peripherals ? Good publicity TI! (Attn:Mike Lunch).

For those of you despairing of receiving 99er Mag, my copy of Issue 4 arrived by airmail from the Publisher on March 15th. Direct airmail subs at present are US$43 for 6 issues, but if they can get enough subscribers they hope to go monthly soon (bi monthly at present). The US postage per copy is over $3 so they are not making a fortune...

99er Magazine is very strongly recommended - this magazine is more than one - it includes a general section, a Computer Aided Instruction section (on loCAItion), LOGO times, and from issue 5 'computer gaming'.

Issue 4 gives news of the goodies to come:

i. most peripherals shortly to be discontinued. A new system based on plug in cards is on the way- simple to use & good looking. Following prices are for US MAINS SUPPLY and are US retail prices - £ prices likely to be near to $- just change sign!
   Expansion box- $200
   NEW RS232 card (now with PARALLEL output) US$140 (old one $165).
   NEW Disk Controller(can run double sided disks)US$200 (old $220).
   NEW Disk drive (s/sided-only one poss)US$300
      ---no price for d/sided drive
      ---2nd and 3rd drives have to be stand alone, only one fits into expansion box.
   NEW Expansion Memory (no changes) US$230 (Old one $290)
   'P CARD' - US$300 - required for PASCAL, PILOT, & others to come- none yet available.

ii. Some very interesting new modules-
   a) Munchman (eg PACMAN) US$40
   b) Mini Memory- a module with 4k memory for YOU to fill (& erase) with 4k of program or data, which retains its memory up to 2yrs! Also adds 32k expansion mem. availability to TI BASIC, permits assembly writing & running without anything other than console, programs can now be 'assembled', and a high-res graphics prog.
   c) Assembler/Editor- module, two disks and 445 page manual!, also coding for 'Tombstone City'(usually $40 as a module)-

The Mini Memory and Editor/Assembler each cost just US$100 - extremely good value compared to any other system.

Because the 99/4A has a re-wired GROM socket, these last two MAY not work on an old 99/4 - I've dropped a line to TI to find out, will report when reply to hand (if they know... they may have to wait until TI USA make the modules available in the UK).

In the USA at least TI seem now to be making a real effort. It is possible that the 99/4 was produced purely to establish the patents, and to a certain extent act as a 'test bed'. The new range of products seems to establish the 99/4A as a major product in TI's range- possibly now to become established- eg less precarious than the 99/4.

Mr Brazier of Cardross has kindly sent me the Addenda to the Extended Basic Manual re Version 110 (I have version 100). There appears to be a tiny error- with my 99/4 and version 100, I have been able to save and load data files using a file name with lower case characters, by concatenating the file name with CHR$. It is true the 99/4 does not have lower case available from the keyboard.

To complete my data- can someone with Version 110 and an Expansion Memory tell me the character for !@P- and +.
Method- switch on,and enter the program line:

```
100 !@P- :: !@P+
```

Now,in Command mode, enter:

```
FOR CT=0 to -20 step -1 :: CALL PEEK(CT,A) ::
PRINT A; :: NEXT CT
```

After ENTER 20 numbers should appear on the screen- please could someone send me a note of them? Hidden in there are the command codes. Ta.

MORE ON MEMORY:
Wondered why the 99/4A has memory locations from decimal -32k to +32k instead of 0 to +64k????
'cos PEEK & LOAD can only address location numbers up to 32k. The negative numbers are derived from ACTUAL memory location s in excess of 32k by subtracting 65536 from the actual location. (64 times 1024 = 65536).

How can the CPU address over 64k (eg all that ROM & RAM- can add up to a lot of memory..) - answer= bank switching which is the reason BASIC cannot use the module's routines...

New memory location: Location -31952 & -31951 store the largest line number.
eg -actual usage: CALL PEEK(-31952,A,B) gives A=247 B=248
now using: LOC= A*256+B-65536 we get LOC=-2056 so we:
CALL PEEK(-2056,A,B) and A= 1 B= 144
now using LINNUM= A*256+B we find that this program has the maximum line number of 400.
Locations LOC+2 and +3 give the memory address for that line.
That will give the delvers something to play with for a while..!

In Summary: -31952/1 point to that location in the LINE NUMBER INDEX which stores the highest line NUMBER and the two bytes following that line number point to the code for that line in memory..... some interesting coding going on here.....

Happy Computing

Stephen

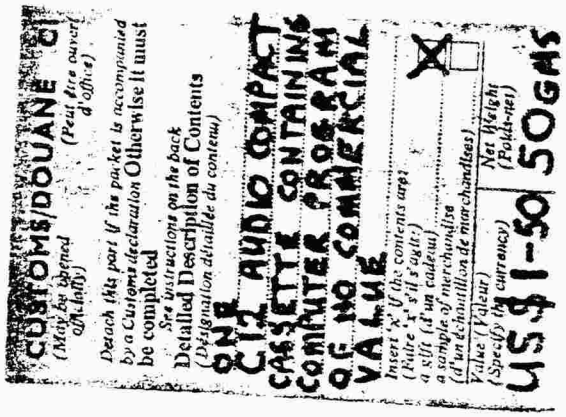
I hope the page numbers & dates help keep these sheets in order - but they will make editting harder. TI HOME of FEB had my myriad of submissions a little out of date sequence (ok- who noticed ?).

PROGRAMS TO SELL ?

Computer & Video Games Magazine pays £10 for each game program printed. You may expressly reserve to yourself the magnetic media right, and can then still send a copy to the club library. State in your documentation: "Magnetic media rights reserved. Copyright 198x by nnnnnnnnnnnn"

Please write to them even if only to let the editor know I'm not the only one with a Texas computer.

The most promising Publisher at the moment seems to by ENTINGER ASSOCIATES INC of
P O BOX 5581 Fort Worth Texas 76108 USA.
Submit your program recorded a few times on cassette with a list IF POSSIBLE and as much explanation as you can provide. Do not forget to attach a customs declaration to the package as follows:



They are interested in high quality software of any type in BASIC or EXTENDED BASIC (remember to state which!)

Your program should be
ORIGINAL
USER FRIENDLY (eg forgive incorrect entries!)

The customs sticker is available from MAIN POST OFFICES & is free.

If you have amassed a whole load of programs of very high quality & feel that you would like some monetary return, I am collecting programs to offer for sale, with low prices and high royalties. The return will not be high - I do not intend to dissuade you from submitting programs to the CLUB! Think in terms of pence per sale! (Does that encourage you to submit to the club library? good!)

TI HOME may be a slightly loose affiliation at the moment - but it is in the interests of every TI owner to fully support it - this way owner pressure can be brought to bear of TI should they do anything really nasty. We do have reports of unhappy treatment of both purchasers & dealers, almost certainly due to insufficient manpower at TI - plus what appears to be a remarkable lack of knowledge.

Please write in - keep in touch - and if you know of any 99/4(a) owner who is NOT a member - a personal invitation often works wonders.

Would you like the club (eg one member of the club....) to offer any services? Club funds ARE low so dont expect too much! (We cannot for instance afford bulk imports of software).

PROGRAM PARTICLES

The following are extracts from various programs, mostly in Extended Basic, which illustrate one or two things in a different manner to the manuals.
Have a look at these and see how they work - and if they are worth using...

```
1.  CALL KEY(0,A,B) :: IF B THEN 100
1b. CALL KEY(0,A,B) :: IF B THEN 100 ELSE 1.
```

```
2. Lower Case characters:  (peculiar capitals actually)..
   FOR I= 65 to 90
   CALL CHARPAT(I,A$)
   B$="0000"&SEG$(A$,1,4)&SEG$(A$,7,4)&SEG$(A$,13,4)
   CALL CHAR(I+32,B$)
   NEXT I
        (Not required for the 99/4A but 99/4
         owners can use it to match character
         sets with the 99/4A)
```

```
3.  1. DEF YES=A$="Y"
    5. ACCEPT BEEP AT(5,5)SIZE(1)VALIDATE("YN"):A$
    9. IF YES THEN 200 ELSE 250
```

```
4.   CALL COINC(ALL,J) ::  IF NOT J THEN 4
```

The above have been kept deliberately short to encourage you to experiment a little.
Sample 1 & 4 use REALTIONAL EXPRESSIONS which return true or false. Branching depends on whether true or not.

Sample 3 is an odd, amusing, useful? use of DEF, again using relational expressions. If A$="Y" then YES takes a 'true' value which leads to the branch to 200.

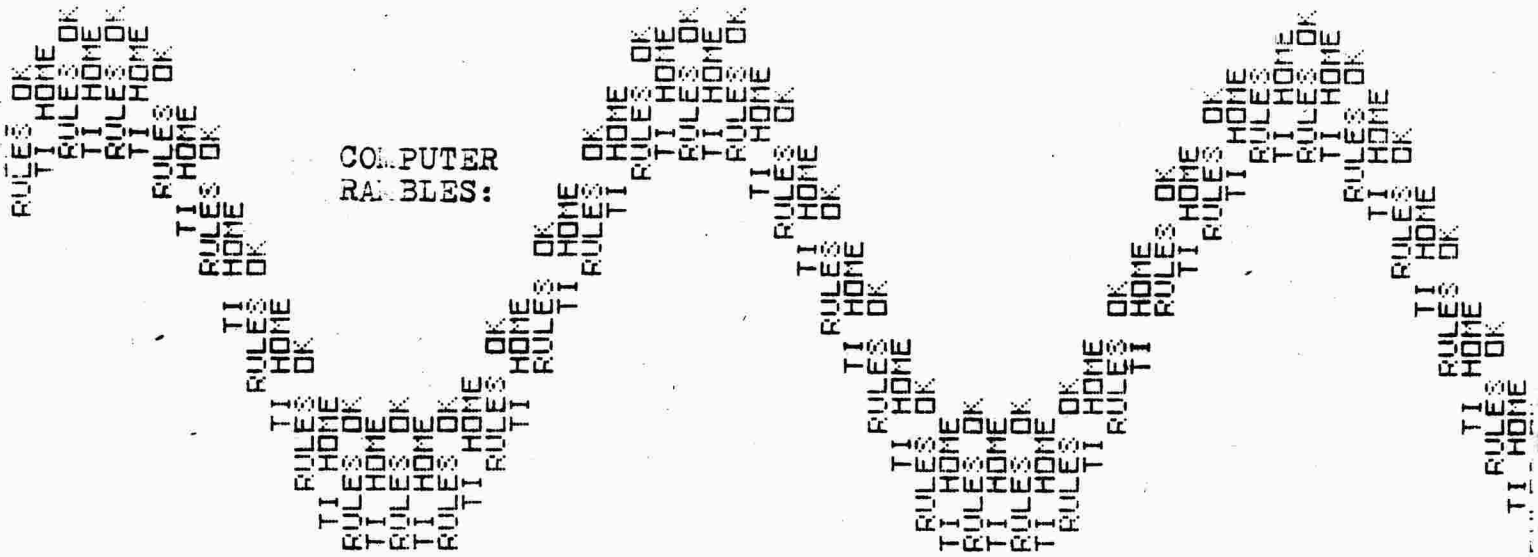
Sample 2 is just a short TI utility.

NEW OWNERS

Are there any features of TI BASIC which puzzle you ?
There are a few areas of the Manual which are difficult for the beginner.
Send me your enquiries- each month/two months,in TI HOME I will answer the most common problems or the most interesting ones. I regret I lack the time to reply to you direct (freely) even if you send an SAE.

Have the discovered anything ? Even a brand new owner can discover things. Write to the club & tell us.

TI HOME would welcome some new contributors- a paragraph up to several pages.
Any short programs ? Any programs for the club library?



9.2.82

Passing through my local BOOTS this lunchtime I was interested to see two assistants struggling to set up a Microcomputer. No -not the TI. This was the BBC!
Now- have TI lost the support of BOOTS ? Or is there some fiendish double marketing experiment in progress? The machines are somewhat similar for a large chain to offer both. Having seen one in the flesh- even if not working - I did not particularly like the look of the BBC micro. The case looks both large and fragile.

To avoid constant repetition can we agree on some abbraviations?
eg TIB= TI BASIC XB= Extended Basic
XM= Expansion Memory TP= Thermal Printer
DSK = DISK CS=CASSETTE PRK=Personal Record Keeping
FAST XB = Extended Basic prog with CALL INIT/CALL LOAD speed up line added.

Any comments?

10.2.82

The struggle was a mite too tough. No microcomputers in Boots Manchester today!

Commercial break---
WANTED: Diagnostic module for short loan or hire- postage refunded. After 500+ hours of use I'd like to give my system a quick check.
WANTED/FOR SALE: Software. 50% royalties max.
99/4 services offered- advice,debugs,listing,bespoke progs.
Drop me a line at above address.

PROGRAMMING HINT

If you have to use DATA/READ, try to only read the data once. If memory permits, it is overall much faster to load your data into a string array for subsequent recall - VERY much faster. If your data is numbers,still use a string array to conserve memory and when actually using it use: P = VAL(ar$(N))

MORE PRK

For some time I have been specifying printer: TP and moaning about the waste of paper....did you know you can respond with TP.U.S.E ? Interesting-character 95 is used with TP but character 28 with TP.U ! If you put a trailing full stop after the E it is ignored...
Fascinating module this. (Personal Record Keeping Module). Using TP.U.S.E gives a pretty neat printout.

COMPUTER & VIDEO GAMES MAG: Program for 99/4 - (2021-see below)
I understand that members with the 99/4A have had a problem with my Extended Basic program published earlier in the year in this magazine.
The problem arises from the different character set of the A-
to debug: CHANGE:

```
CALL CHAR(100,"96FEBA3838BAF
EBA")          INTO:

CALL CHAR(100,"96FEBA3838BAF
EBA"&RPT$("0",48))
```

This program has been submitted tc the club library in amended form as CARRACE.
My apologies to everyone who had problems!

ZERO ZAP HIGH SCORES: This game allows only for scores up to 999. Go over that -if you can- and the score 'wraps around' so that you can have a 'Best Score' of (apparently) 112 which beats a score of 500! The module remembers the thousandsb but does not display them. The 'Attract Mode' where it plays all on its own is MUCH better than throwing a coin in the air - just bet on which 'player' is going to win! Completely random.

INDEPENDANT SOFTWARE:
Two cassette programs for the 99/4 and 99/4A, in TI BASIC are available at £9 each from:
Work Force, 140 Wilsden Ave, LUTON, Beds.
These are "The Cube" and "Quadcube(tm)" - the latter a 4x4x4 cube- impossible to manufacture but easy to use on a computer. The programs are on cassette,ONLY.
I have seen 'Quadcube' and can recommend it. If you have a disk system the program needs a lot of compacting- eg replace common numbers with variables and check the call keys with 'on 1+Pos(...)' etc.
If you are a cube twiddler, you will like these programs.
The notation is simple and easy to use.Documentation of good quality.

2021 update- the game referred to at the top was called SPEED RACE and it appeared in Computer and Video Games Magazine Issue 3, January 1982, on page 32.

AMEND LINE 130

C&VG did not print listings very well, but a scan of the page is added as a supplement to this magazine scan.
The game can be found in the TI Gamebase, but it appears to be the uncorrected TI99/4 version, requiring the correction noted above.

IMPORTING PROGRAMS FOR THE 99/4 YOURSELF:

Although software is in short supply in the UK, there are many suppliers in the USA, and you can find discounts even on TI Software.

I have now received programs from two US suppliers and can report on the procedure -

When you order, always send your payment by Bank Draft, in U S Dollars, drawn on an American bank (in New York or Washington is preferred). Remember postal rates are quite high - if the supplier does not quote a foreign rate you must ask. (Some take Visa & Mastercharge).

Ensure the supplier can handle foreign orders - if he has never sent packages abroad before he may neglect to use a Customs declaration & your package is then subject to seizure!

To keep post to a minimum make your wishes very plain on what you want him to do if part of your order (or all of it!) is out of stock - how long he is to hold it. If he has to send two parcels, the post will cost that much more!

US retailers have as much difficulty obtaining TI software as we do!

'Always ask for the package to be sent airmail- it costs a lot but the chances of loss are less. Ask for it to be insured.

When your package arrives in this country, you MAY receive a letter from HM Customs asking for a copy invoice (send copy advert & copy order if you have no invoice) and asking for details of the computer program - what it is for, what it does, if it is for resale & so on. You must return this with a post office clearing fee (£2.50 on parcel post).

In due course the postman will knock on your door, with a parcel, which may have postage due stamps on it, or a TO PAY rubber stamp. You must pay the postman the Duty and VAT if these have been assessed, IN CASH (no cheques!). He then hands you your parcel.

How do costs compare? I sent an order to Microcomputer Corporation, as below, and when all costs have been included and split in proportion between the programs, the cost was as follows (inc duty, vat, post, PO Fee, etc):

Program:	Inclusive price from USA:	Current UK advertised price:
ZERO ZAP (module)	£13-50	£19.95
Teach Yourself Extended Basic (cassette)	£13-50	£29.95
PROGRAMMING AIDS two (disk)	£16-50	£24.95
three (disk)	£13-50	£19.95
TOTAL:	£57-00!	£94-80

SAVING = £37-80 !!

IMPORTING (2):

Microcomputers Corporation have a new address:
34 Maple Avenue P O Box 8 Armonk New York 10504 USA .

The five programs I ordered & received from them arrived very well wrapped and loaded first time- brief reviews-

1 Teach Yourself Extended Basic-

Interesting - I now know that Assembly Language programs reside in memory locations 8192 to 16383 (Hex 2000 to 3FFF), which contain numeric values up to 255.

IF...THEN..ELSE: Did you know that ELSE can appear AFTER a double colon after the IF? I didnt...
eg IF A=B THEN PRINT "HI" :: C=5 :: R=TEST :: ELSE PRINT "HMM"
This is quite powerful!

Also learned that FOR NEXT can be used in command mode- eg no program required!
eg FOR CT=1 to 1000 :: PRINT "HELLO "; :: NEXT CT

I have a feeling this material could have been presented in the reference manual though Admittedly, sprites are well presented in this format.

2 Programming Aids 2:

Sorting routines - ONLY for FIXED length DATA files on DISK.

3. MARKET SIMULATION:

A fairly serious business simulation, based an a New York University program of 1972 vintage.
Two companies produce the same product, with the same fixed costs . Variables to be input are:
Number of units produced, selling price, and amount spent on advertising.
The winner is the first to reach a certain amount of assets or the one left after the other player becomes bankrupt.
Certain random events take place affecting one or both players- eg vice president runs off with 2000 dollars...! but the balance of the program is one of skill.

4. ZEROZAP - a games module from Milton Bradley.

This is largely a visual/sound entertainment, as there is very little player interaction and relies almost entirely on chance. It is of interest however for the sheer speed of the program. There are a lot of things happening very quickly. You 'launch' an arrow from the bottom of the screen and this bounces around targets, scoring as appropriate. It may fall back to the ground or if it hits a ZERO it is knocked to the ground. The score actually COUNTS UP -very very quickly-eg all intermediate numbers appear.

The Zero's appear to be sufficient to keep your average score at 100 per arrow, but you can score as much as 250 on rare occaisions. There is lots of sound- if you find the music tiresome after a while, press any key and it will cease.

There are three standard layouts, and you may create your own- which can be saved to tape.

The only resemblance of THIS game to pinball is to those

cont...

zero zap cont/..

pin ball machines which will not give you the opportunity to FLIP the ball! but keep it shuttling between bumpers.

5. Programming Aids 3 (Extended Basic & Disk Drive & Printer):

a) it lists the items in your program as shown below
b) it enables blocks to be deleted or resequenced

Sample program list:
(excerpt only! much longer...)

```
CROSS REFERENCE OF
AUTO CHAR DEF ON 200282
        AUTO CHAR DEF/1

 NUMERIC ARRAYS
---------------

B()
        440   510   520   670

 STRING VARIABLES
-----------------

HEX$
        480   530
M$
        160   200   240   280
  490   530   550   630
        670   700   770   780
  820   830

 NUMERIC VARIABLES
------------------

C
        350   360   440   450
  460   870   890   900
        910   960   970   980
CODE
        780   790   830   840
HIGH
        520   530
I
        770   780   790   800
  820   830   840   850
KEY
        370   390   420   430
  440   450   730   750
        880
LOW
        510   530
R
        310   320   330   340
  360   440   450   470
        500   510   520   540
  570   580   590   870
        920   930   940   990
 1000  1010
STATUS
        370   380   730   740
X
        160   260   610   650
  790   840
Y
        170   210   250   290
  600   640   680   710
        790   840

 BASIC KEYWORDS
---------------

CALL
        120   130   140   150
  320   360   370   450
        550   560   580   730
  790   840   870
FOR
        310   340                 500
```

BUGS:
Presentation was not good on the thermal printer- the following lines were amended:

```
310 PRINT #2:"CROSS REFERENCE OF
 "!T$&" ON "&D$
520 PRINT #2: : :" ";H$(T);SEG$(
"                    ",1,LEN(
H$(T))+1);" :"
520 PRINT #2:" ";N$:"     ";
700 PRINT #2:"     ";
860 PRINT #2:"  ";
880 PRINT #2:TAB(10);T$;"/";STR$
(P)
```

This program was reviewed in 99er Magazine Issue 3

NB: If you have the Expansion Memory, you may run all segments in Extended Basic- resequence the first segment and add:CALL INIT :: CALL LOAD (-31878,0) then add an extra line at the end of each segment to RUN the next.

Example: at the end of program CREF replace STOP with:
RUN "DSK1.CREFPRINT"
There is adequate room on the disk to take all the needed program handling. You only need your original program disk to load the console, then SAVE, MERGE to the Programming Aids Disk. The program will delete the files when it has finished with them.

((There is a possible error condition with overlength program lines))

THE PUZZLE & MYSTERY OF THE PERSONAL RECORD KEEPING MODULE

We have already seen in TI HOME that when using BASIC with the PRK module inserted, some extra sub-programmes are available, of which two can be of practical value:

CALL A = ACCEPT AT
CALL D = DISPLAY AT

The others- CALL S, CALL L, CALL G, CALL H, CALL P do not appear to be of use, generating some different error messages.

THE PUZZLE: What are these 5 CALLS?

If a= accept and d=display, then the other letters may also be related to their function.

If you have a disk drive, you may have noticed that PRK data is saved NOT in file format, but as a single program! If you try to run the program in BASIC, it does not exist as far as the computer is concerned! Load it into Extended Basic, and after a long pause you will return to the title screen but with redefined graphics! (Operation is not affected- select 1 or 3 and you can load a proper program).

Therefore the PRK data has to be stored -in program format - in the 'program' sector of machine memory - or assembled into program format when it is saved.

CALL S & CALL L have the same format:
CALL S(or L) (string,number).
Could these be Save and Load ?with string relating to file and number to page ?

CALL G & CALL H have the same format as each other:
(number,number,number,number.....) - at least four numbers but may be more.
Further, the first number must be 0-3. There are just four types of data with PRK- Char, Dec, Int & Sci - related???
So if we start with a single definition of one data field, the remainder may be the data- but in numeric format only, presumably with ASCII codes for CHARs and command codes as appropriate.
But what could G & H stand for or do? Are they HCHAR & GCHAR -unlikely with D available. Perhaps Get & H...?

With S & L if NO pages are held, any page number you enter would quite correctly be too big- the error message generated.

If G & H handle data in memory, with Basic selected they would assume memory was full, hence Not Enough Memory.

P ? How about the number of pages?

MYSTERY: Why do we get these error messages in BASIC at all, if we were never intended to use (or see) them? It would have been sufficient to use 'bad name'.

CASSETTE TROUBLE?

Do you have any trouble with your cassette recorder ?

The following remedies may help you:

REMOTE CONTROL NOT WORKING:

a) Try easing the plug out one or two mm. There can be quite considerable differences in the construction of these 2.5mm sockets, and your tape recorder may be expecting a slightly shorter plug.

b) Try reversing the polarity of the plug- cut the wire to the plug and swop the conductors over.
Your computer uses electronic switching,and in the absence of any agreed standard there may be a clash between your computer and your tape recorder (it wont cause any harm though - just stop the remote working!).

c) If you have difficulty saving and loading your own programs, listen to the part of the tape between the computer starting the tape running and the beginning of the opening tone. If there is an appreciable background noise present, picked up from the computer or your television, try inserting a resistor into the microphone lead of your recorder - start with say 47k or 39k. & work downwards.

HINT: When checking or loading, if the volume is too high you will get 'NO DATA' - you will usually get 'ERROR IN DATA' if it is too low, unless it is VERY low when again you get NO DATA. The setting may be sensitive on your recorder.
Keep the tone control turned up!

Review.....

s t a r l o r d

There is no 99/4 involved - but our interest is in general computing as well as our own small corner ?

s t a r l o r d

is a play-by-mail fantasy game set in a distant galaxy where a once great empire has collapsed and all the star lords are seeking to usurp the empires throne

it is a game played with 20k of data in machine code and 20 programs in basic
played on a pet
and using a COLOR printer

Each player has a base star - he builds up his fleet by capture, by building and by purchase.

the galaxy is huge - and each player can only see a circle of the galaxy 7 astrals in radius.

the aim is to capture the throne star and become
EMPEROR
then you have to keep the throne - there are up to 49 other players trying to dethrone you. The Emperor has a galaxy-wide map.

You drop out of the game either
a) by not paying your dues or
b) by your command ship having to retreat to base- and finding it owned by some other starlord.

((How big is a galaxy? in this case it has a diameter of 128 astrals))

There are 14 types of star each with its own purpose. Even black hole gateways. There are four battle orders and four defense orders. There is NO random element in this game.

Rules are simple - orders given are simple - entry of your orders into the computer is simple!!

Starlord began in April 1981. There are now over 250 players in 6 separate galaxys.(As at Dec 81).

COST: £1.25 for first 2 turns & rule book,thereafter £1.25 per turn. Turns are 2 weeks apart. (An international game has turns 4 weeks apart & costs the same or US$3.75 per turn)

s t a r l o r d is operated by Mike Singleton (to whom cheques payable) & operates from 1 Rake Hey Close Moreton Wirral Merseyside L46 6EW

OK. Now - how about a game for us 99/4 owners eh? Mikes original idea was for game turns exchanged on cassette - real time battles - 3d maps - ideas to consider. Starlord communicates purely by printout - so you do not need a pet to join in.

RAMP-DOWN TONES

IN T I BASIC

```
10 FOR T=1 TO 10
11 W(T)=110+T*25
12 NEXT T
20 F=300
60 FOR C=1 TO 1000
62 F=W(INT(RND*10+1))
100 FOR N=.55 TO .15 STEP -.05
120 CALL SOUND(-1000,F+N*F,0)
130 NEXT N
200 NEXT C
```

32k ram required for the next two:
plus Extended Basic

```
100 CALL INIT :: CALL LOAD(-3187
8,0):: CALL CLEAR
110 FOR T=1 TO 10 :: W(T)=100+T*
24 :: NEXT T
120 FOR C=1 TO 20000 :: F=W(INT(
RND*10+1)):: FOR N=.6 TO .1 STEP
 -.08 :: CALL SOUND(-700,F+N*F,0
):: NEXT N :: NEXT C :: BREAK
130 END
```

```
100 CALL INIT :: CALL LOAD(-3187
8,0):: CALL CLEAR
110 F=140 :: FOR T=1 TO 10 :: W(
T)=100+T*24 :: NEXT T
120 FOR C=1 TO 2E9 :: CALL KEY(0
,A,B):: IF A>20 THEN F=MAX(110,A
+A+.2-300)
181 FOR N=.6 TO .1 STEP -.08 ::
CALL SOUND(-700,F+N*F,0) :: NEXT
N :: NEXT C :: BREAK
190 END
```

EXTENDED BASIC

```
100 REM ********************
101 REM *                    *
102 REM * HEX TO DECIMAL     *
110 REM *    CONVERSION      *
120 REM *       FROM         *
121 REM *  MICROCOMPUTER     *
122 REM *  PRINTOUT  2/82    *
123 REM *                    *
130 REM ********************
132 REM
140 INPUT "ENTER HEX NUMBER:":A$
 :: J=0 :: FOR I=1 TO LEN(A$)::
K=ASC(SEG$(A$,I,1))-48
150 IF K<10 THEN 160 :: K=K-7
160 J=J*16+K :: NEXT I :: PRINT
J: :: GOTO 140 :: END
```

```
 1 REM  HOW A COMPUTER SHOULD
 2 REM  LOOK WHEN IT IS
 3 REM  WORKING!
 4 REM  COURTESY
 5 REM  VOYAGE TO THE BOTTOM
 6 REM  OF THE SEA & STARTREK
 7 REM
 8 REM  BY S SHAW  1982
 9 REM  IN TI BASIC
 10 REM  :::
 50 RANDOMIZE
 100 FOR I=1 TO 16
 110 CALL COLOR(I,I,I)
 120 NEXT I
 130 FOR I=1 TO 24*28
 140 PRINT CHR$(RND*127+31);
 150 NEXT I
 160 REM
 170 RAN=INT(RND*16+1)
 180 RAN2=INT(RND*16+1)
 190 CALL COLOR(RAN,RAN2,RAN2)
 200 CALL COLOR(RAN2,RAN,RAN)
 210 CALL HCHAR(RND*23+1,RND*27+1
,RND*120+32)
 220 CALL HCHAR(RAN,RAN2,RND*120+
32)
 230 REM
 240 CALL HCHAR(RND*23+1,RND*27+1
,RND*120+33)
 250 CALL HCHAR(RND*23+1,RND*27+1
,RND*123+33)
 260 CALL HCHAR(RND*22+1,RND*27+1
,RND*120+37)
 500 GOTO 160
```

Condensed Format Code;

In 99er Issue 3 we were given a list of Condensed Format Codes-
If you use extended basic & expansion ram, you can see that your program commands are stored using this code- eg the Basic word 'PRINT' is stored in one memory location as code 156.
Numbers are stored as 200 N 1 2 3 4 5 where N is the number of numbers (this is numbers in your program list eg a=12345 or goto 12345.) The minimum storage space for a number is 3 bytes - hence the saving if you use variables.

There were some numbers whose use was unknown in 99er- here are a few, courtesy of the Programming Aids 3 program:

169 -listed as RUN - is considered by PA3 to be 'library'.

The following are NOT available in Extended Basic, but the codes are allowed for in PA3:

171- INTEGER	226-UPRC$	227-STATUS	228-TIME$
229-DAT$	230-INTG	231-ALPHA.	

Attention 99/4A owners- when using ACCEPT AT with a validation command (eg VALIDATE(UALPHA)) can you use VALIDATE(ALPHA) ? or do you have UPRC$?

As these commands are in the code table, could they have been intended for use with Extended Basic, or has the code table been taken over from some other language ???

LINE NUMBERS are dealt with as two bytes (referrals to line numbers in your prog are dealt with as above however) -
the line number = byte 1 * 256 + byte 2.
Each line in your program is indexed- two bytes for the line number and two bytes for its location - so keep your lines to a minimum!
Remembering that REM is stored as one byte, the following line: 1 REM takes up 7 bytes!
four bytes in the index, one byte for REM, one byte to mark end of line, and one byte to indicate length of line.

With this knowledge you should be able to reduce the memory usage of 'tight' prgrams.

(NB: The double colon :: is one byte-code 130).

HOW A PROGRAM IS STORED in memory

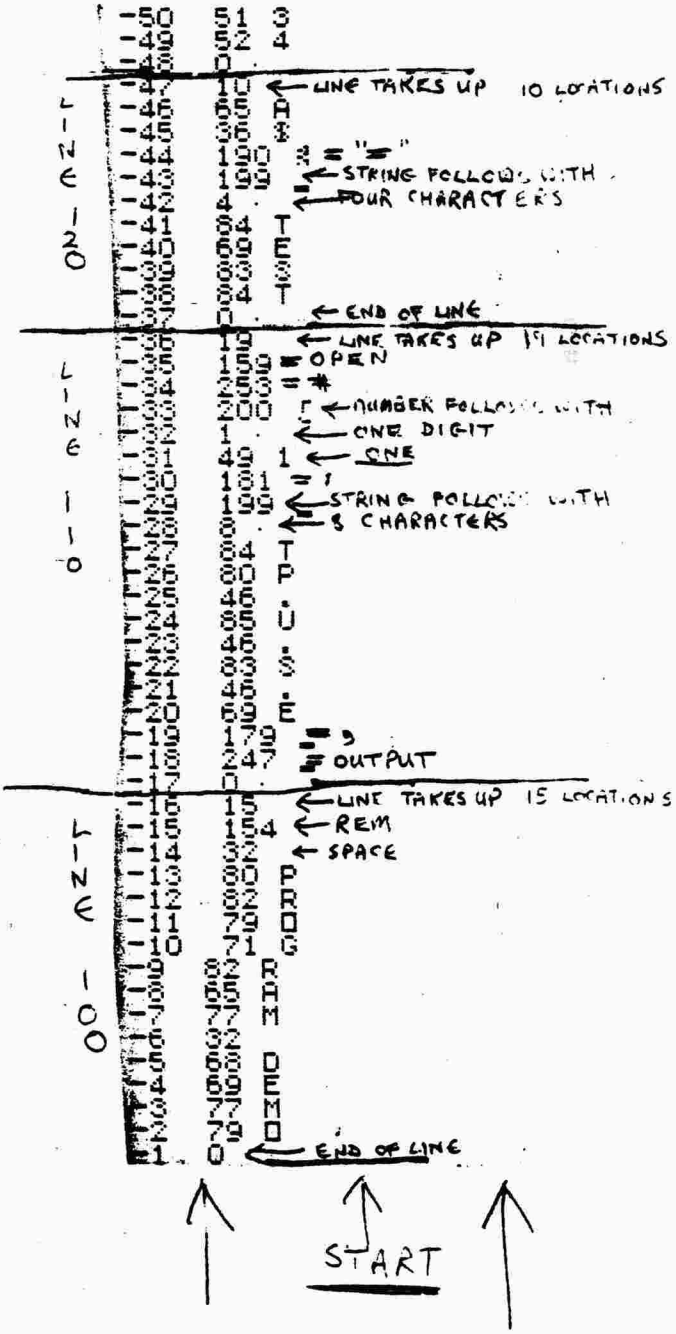
(These memory locations apply ONLY with Expansion RAM)

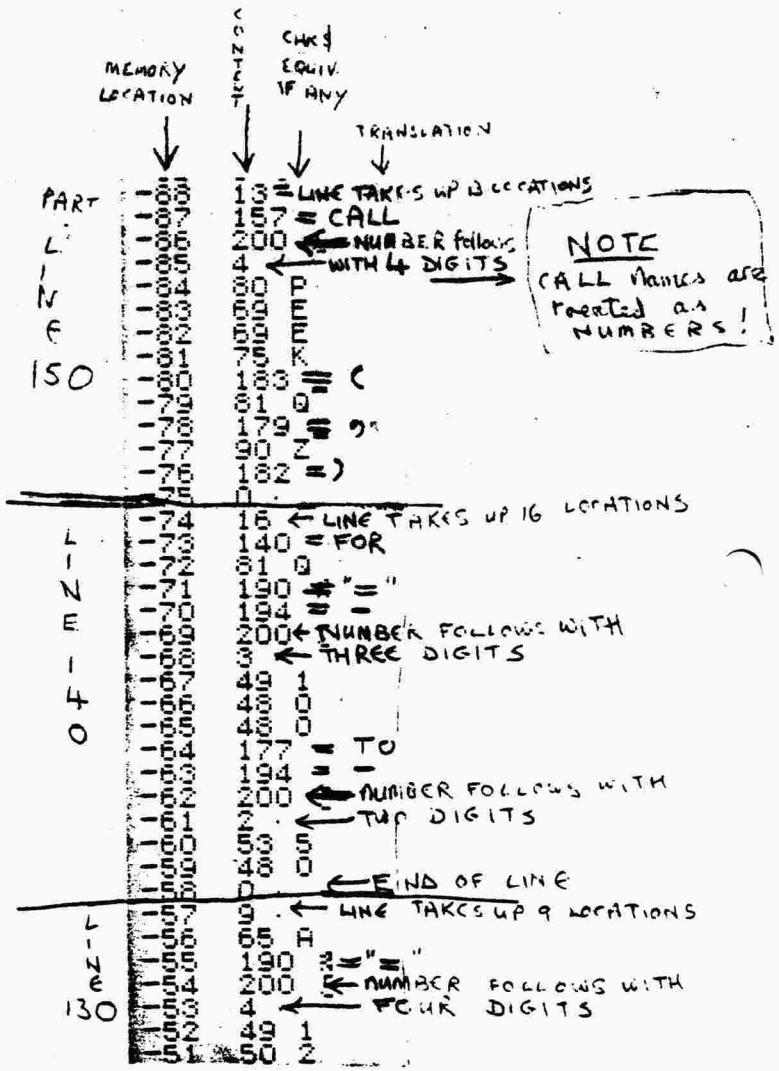
PROGRAM:

```
100 REM PROGRAM DEMO
110 OPEN #1:"TP.U.S.E",OUTPUT
120 A$="TEST"
130 A=1234
140 FOR Q=-100 TO -50
150 CALL PEEK(Q,Z)
160 CALL PEEK(Q+50,X)
170 PRINT #1:Q;Z;CHR$(Z),Q+50;X;
CHR$(X)
180 NEXT Q
190 END
```

FIRST FEW MEMORY LOCATIONS:
(Program stored from 0 to -24k):

Here is how lines 100 to 140 are stored in the 32k ram:

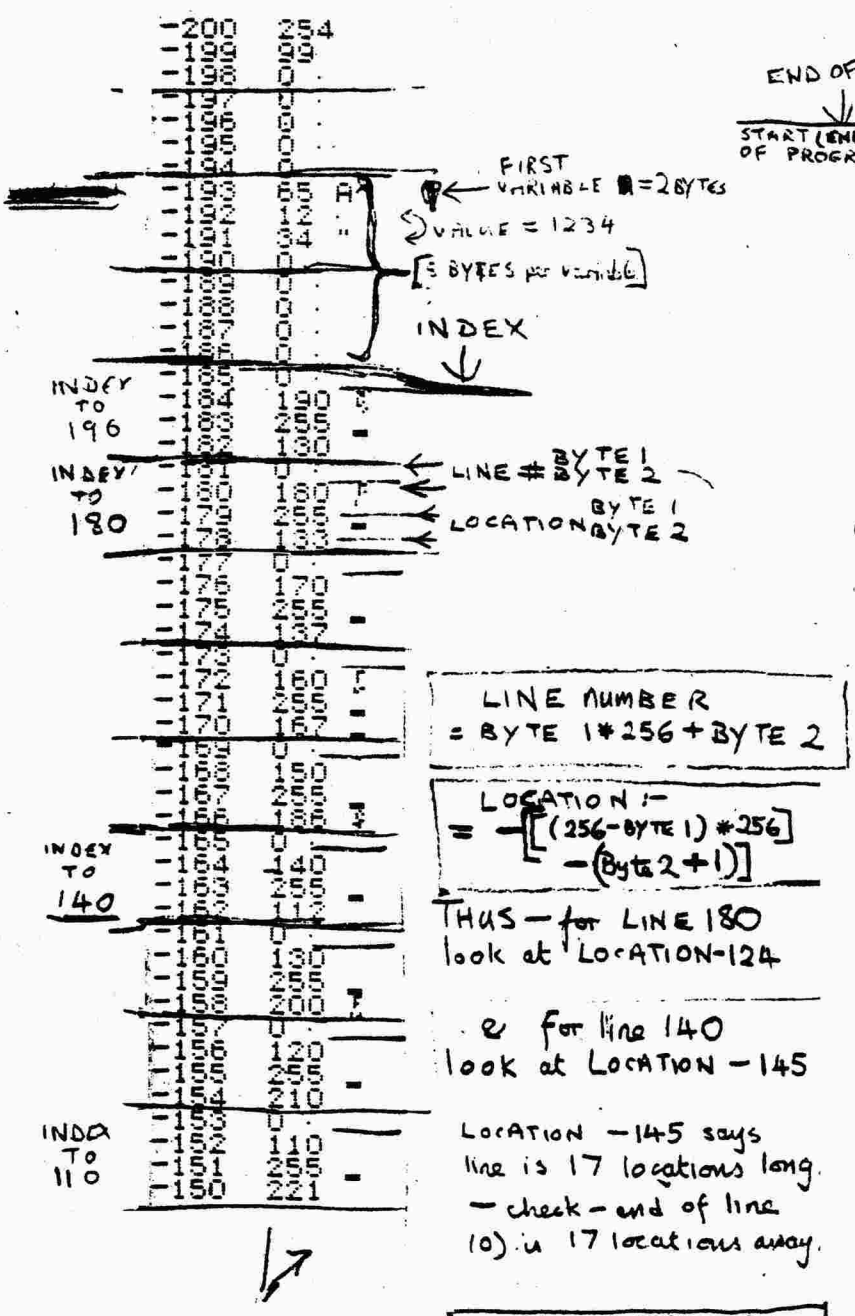




Memory use after line 140 has been edited:

```
100 REM PROGRAM DEMO
110 OPEN #1:"TP.U.S.E",OUTPUT
120 A$="TEST"
130 A=1234
140 FOR Q=-200 TO -150
150 CALL PEEK(Q,Z)
160 CALL PEEK(Q+50,X)
170 PRINT #1:Q;Z;CHR$(Z),Q+50;X;
CHR$(X)
180 NEXT Q
190 END
```

Program line 140 now moves from memory location -74 to -175. Program lines are stored in the ORDER INPUT not line number order- hence the need for an INDEX - memory locations -184 to -146:



LINE NUMBER = BYTE 1 * 256 + BYTE 2

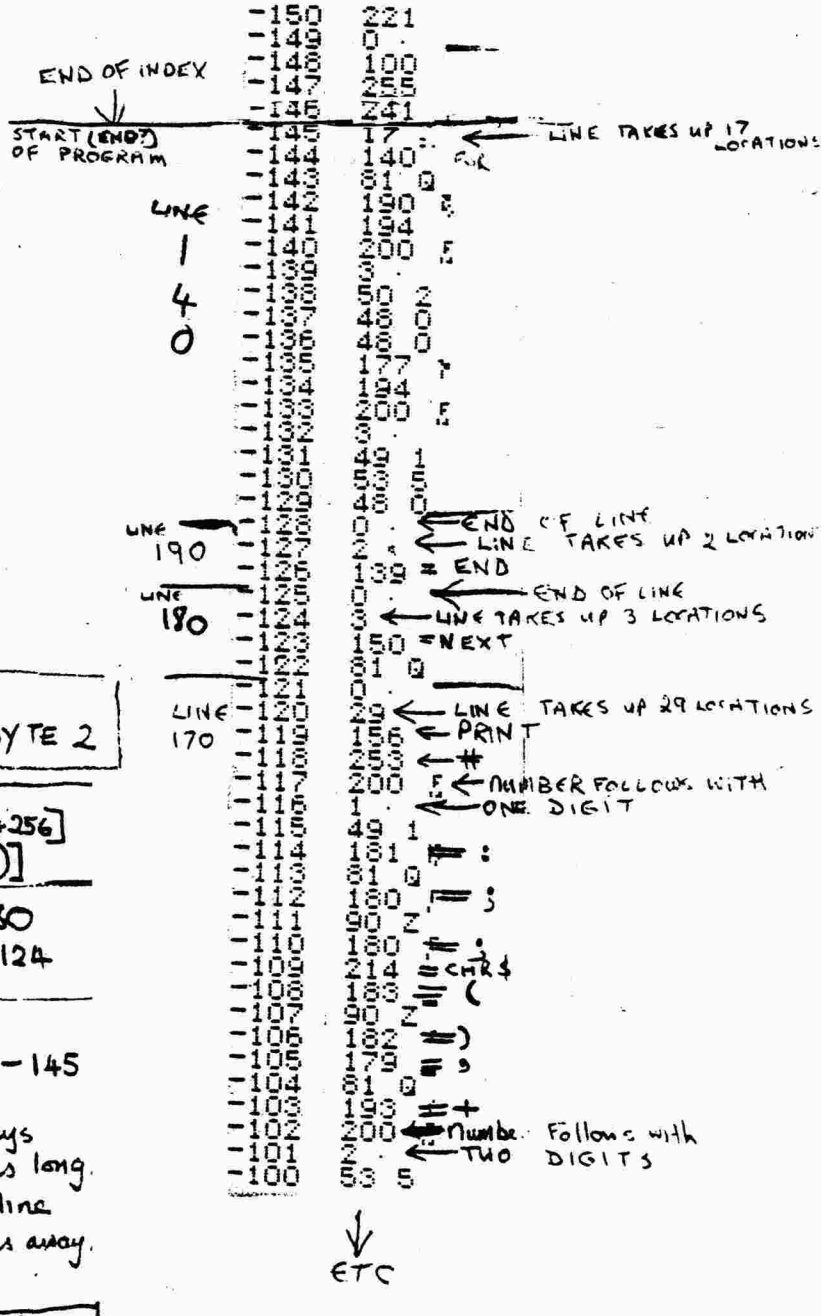
LOCATION :- = -[[(256 - BYTE 1) * 256] - (Byte 2 + 1)]

THUS - for LINE 180 look at LOCATION-124

& for line 140 look at LOCATION -145

LOCATION -145 says line is 17 locations long. - check - end of line 10) is 17 locations away.

LOCATION IS OFFSET BY ONE >COS -2 IS LOWEST POSS.



(Program lines 100-130 have the same memory locations after editing line 140. Lines 150-190 change memory locations.)

------> continued

Program storage 3:

The variable number stored at -193 to -190 needs a little more detail- the name of the variable is NOT included.

Lets see some examples-

we have seen that VARIABLE= 1234 gives us:	VARIABLE =7 gives us:	VARIABLE =499931 gives us:
65 (A)	64 (@)	66(B)
12	7	49
34		99
		31

SO-
The first data is the number of bytes the number is stored in, coded so that one byte=64, two bytes=65, 3=66 etc

The numeric variable is divided into 2-digit lengths and stored in order.

IF MORE than one numeric variable is set, they each take up 8 bytes,stored in this fashion- eg:
first variable: 64.7.0.0.0.0.0.0
2nd variable: 66.49.99.31.0.0.0.0

As with program lines, the last one set has a higher negative location number than the previous one.

Further information is stored in lower memory locations, but I have not yet cracked the coding!! Presumably the location holding the value of A is tied into A somewhere!

The coding of the length of the numeric variables is very interesting! Why should 64=length 1 !

String variables are kept in a different section of memory not yet traced.

WITH MEMORY EXPANSION you can use CALL LOAD to place values into these memory locations. This means your program can permanently rewrite itself! You can also write in interesting self-destruct codings!

POSTSCRIPT

Well, there you are, yet another newletter has come to an end. Think about it, and enter your opinions on the questionnaire and send it back to me. You must realise that I started TIHOME over 18 months ago and have been running it very much in isolation ever since. The first year was really difficult as I was running it out of my own pocket, however, things have improved since then, so come on, let me know what you want and need and I will attempt to supply the requisites.

Do not be worried or disturbed about any of your problems. When in doubt ring TIHOME. I will always try to give you a sensible answer. If I can't give you a sensible answer I will include your moan in my next nag to TI thamselves.

TI have a great respect for TIHOME. Use this respect, if you have a problem get in touch with me and I will pass it on. Be assured, I always get an answer. If the problem is really tragic I will put you in touch with TI and you can talk to the TOP people.

I want to start a Computer Pen-Friend spot in the newsletter. If you want a pen-friend, let me know and I will publish your name & address in a dedicated section. Anyone answering a request for a pen-friend should accept that they may or may not be answered.

Another thing I want to do is to organise meetings for members in different districts, so be sure to fill in the questionnaire and give me some idea of who I can contact to organise district meetings.

I am always willing to use offers of help with the running of TIHOME. If you are interested, please let me know with a note of what you think you can do. Please realise that TIHOME is a non-profit making organisation and so cannot afford to pay anyone, including me, as 100% of the subs go to running the organisation. In the future, perhaps, TIHOME can pay wages and fees, but the membership will have to rise a lot above 180. Anyway, think about it!!!

A last tip for you beginners with a 4A, please remember that OLD CS1 and SAVE CS1 must be input with the ALPHA CAPS key locked down. The cassette interface does not recognise Old or Save Csl. You understand?

Tidings has as yet not taken adverts, however, I am now prepared to take adverts at the rate of £5 per A4 sheet per one issue, or proportionally less per area used per issue. Personal adverts will be accepted at the rate of 20p per line of 90 characters per issue. Send any adverts in with the correct money detailing the issues required, remembering that there are 6 issues of each volume each year, and each volume begins in February of each year.

If you send programs for inclusion in the Library, please remember that they will probably be distributed world-wide. Do not include your name & address or any request for money unless you are prepared to deal with the problems of international exchange, or are prepared to receive letters from Australia or Hongkong, etc.

God bless, and the best of computing luck,

Paul

Published by:- TIHOME, Paul Dicks, 157 Bishopsford, Morden, Surrey, England.
Registered No. 2778359

QUESTIONNAIRE.

1. Do you object to your name and address being published?

2. Do you think a Children's Page would be of use in Tidings?

3. Do you think a Beginner's Page would be of use in Tidings?

4. Are there any services that TIHOME can give you that it does not already do?

5. What is your biggest complaint against TI?

6. Are you successful in contacting TI in Bedford.

7. Are you prepared to use TIHOME as your go-between in dealing with TI in Bedford?

8. Do you have any suggestions for articles in Tidings?

9. Do you have any particular problems with hardware?

10. What did you think the 99/4(A) could do for you when you bought it? Has it lived up to your expectations?

Please fill in this questionnaire and return as soon as possible to:-

TIHOME
Paul Michael Dicks MIDPM
157 Bishopsford Road
Morden
Surrey

T.I. HOME

SUBSCRIPTION REMINDER

MY RECORDS SHOW THAT I HAVE NOT RECEIVED YOUR SUBSCRIPTION OF £9.50 (CHEQUES MADE PAYABLE TO T.I. HOME).

I WOULD BE GRATEFUL IF YOU WOULD SEND ME YOUR REMITTANCE AS SOON AS POSSIBLE.

THANK YOU.

PAUL DICKS.

```
10 CALL CLEAR  ! amended for XB V110 on a 4A- 2021
20 PRINT "SPEEDRACE":"COPYRIGHT 1981":"BY STEPHEN SHAW":"10,ALSTONE
ROAD,STOCKPORT":"CHESHIRE SK4 5AH": : :
30 PRINT "USE S & D TO MOVE ":"LEFT & RIGHT":"    ":"USE KEYS 1,2,3,&4 TO":"SELECT
GEAR"
40 PRINT "TIME & DISTANCE ARE ":"DISPLAYED.":"DISTANCE SUFFERS IF
YOU":"CRASH"
50 PRINT "PRESS ANY KEY TO CONTINUE"
60 CALL KEY(0,V,M)
70 IF M<1 THEN 60
80 CALL SCREEN(2)
90 FOR X=1 TO 100 :: NEXT X
100 CALL CLEAR
110 CALL MAGNIFY(2)
120 M=1
130 CALL CHAR(100,"96FEBA3838BAFEBA"&RPT$("0",48))
135 CALL CHAR(110,"5A5A5A5A5A5A5A5A")
140 CALL CHAR(105,"FF000000000000FF")
150 CALL SCREEN(4)
160 CALL SPRITE(#6,110,13,80,1,90,0)
170 CALL SPRITE(#7,105,13,75,17,90,0)
180 CALL SPRITE(#8,105,13,70,30,90,0)
190 CALL SPRITE(#9,110,13,65,1,90,0)
200 CALL SPRITE(#10,105,13,60,17,90,0)
210 CALL SPRITE(#11,105,13,55,30,90,0)
220 CALL SPRITE(#12,105,13,50,1,90,0)
230 CALL SPRITE(#13,105,13,45,17,90,0)
240 CALL SPRITE(#14,105,13,40,30,90,0)
250 CALL SPRITE(#15,105,13,85,145,90,0)
260 CALL SPRITE(#16,105,13,80,157,90,0)
270 CALL SPRITE(#17,110,13,75,176,90,0)
280 CALL SPRITE(#18,105,13,70,145,90,0)
290 CALL SPRITE(#19,105,13,65,157,90,0)
300 CALL SPRITE(#20,105,13,60,173,90,0)
310 CALL SPRITE(#21,105,13,55,145,90,0)
320 CALL SPRITE(#22,105,13,55,157,90,0)
325 CALL COLOR(8,3,4)
330 CALL SPRITE(#23,110,13,45,173,90,0)
332 CALL VCHAR(1,8,140,216)
333 CALL COLOR(14,12,12)
334 CALL VCHAR(1,7,95,24) :: CALL VCHAR(1,17,95,24) :: CALL
CHAR(96,"5555555555555555")
340 FOR CT=1 TO 4
350 CALL SPRITE(#CT,100,CT+6,CT*47-45,93-CT*8,0,0)
360 NEXT CT
370 CALL SPRITE(#5,100,16,160,74,0,0)
400 CALL SOUND(-1000,-2,30-7*SPEED)
410 CALL COINC(ALL,D) :: IF D<0 THEN GOSUB 900
420 CALL KEY(0,A,B) :: IF A=ASC("S")THEN CALL MOTION(#5,0,-10)
```

```
430 IF A=ASC("D")THEN CALL MOTION(#5,0,10)
440 IF A<30 THEN CALL MOTION(#5,0,0)
450 CALL COINC(ALL,D) :: IF D<0 THEN GOSUB 900
460 IF A>48 AND A<53 THEN SPEED=(A-48)/3
461 CALL COINC(ALL,D) :: IF D<0 THEN GOTO 521
470 T=T+1 :: S=S+6*SPEED :: DISPLAY AT(10,18)SIZE(10):STR$(S)&"  "&STR$(T)
480 CALL COINC(ALL,D) :: IF D<0 THEN 521
490 IF T/5=INT(T/5)THEN M=-M
500 CALL MOTION(#1,SPEED*40,M*5,#1,SPEED*40,M*5,#1,SPEED*40,M*5,#1,SPEED*40,M*5)
501 CALL COINC(ALL,D) :: IF D<0 THEN GOSUB 900
520 GOTO 400
521 GOSUB 900
522 GOTO 400
900 CALL SOUND(-900,-6,0)
910 CALL MOTION(#1,0,0,#2,0,0,#3,0,0,#4,0,0,#5,0,0)
920 SPEED=1/3
930 S=S-50
931 IF S<0 THEN S=0
932 CRASH=CRASH+1
933 IF CRASH=15 OR T>200 THEN 2000
935 M=1
936 T=T-(5*(T/5-INT(T/5)))
940 FOR CT=1 TO 4
950 CALL SPRITE(#CT,100,CT+6,CT*47-45,93-CT*8,0,0)
960 NEXT CT
965 SPEED=0
970 RETURN
2000 CALL CLEAR
2010 PRINT "YOU HAVE TRAVELLED A":"A DISTANCE OF ";S
2020 PRINT "AND HAD ";CRASH;" CRASHES!"
2030 IF S>500 THEN PRINT "YOU ARE NOT A BAD DRIVER"
2040 IF S<100 THEN PRINT "YOU SHOULD NOT BE ON THE":"ROAD"
2050 PRINT "TO TRY AGAIN,ENTER 'RUN'"
```