free copy

TI-LINES

Volume 1, Issue 1, APRIL 1st., 1984

TI-LINES is produced and published by:

Peter G. Q. Brooks
29 Kestrel Crescent
Blackbird Leys
OXFORD OX4 5DY

Telephone OXFORD 717985 (after 7 p.m.)

Every effort is made to ensure that the information given in this newsletter is correct. The publisher cannot be held responsible for any inaccuracies.

TI-LINES is also available to blind or partially-sighted Users on audio cassette - contact the publisher for further details.

Contributors to TI-LINES should ensure that their material is submitted either as typed copy or in clear hand-writing, where it will be subject to limited editing and retyping.

Submissions should arrive by the first day of the month prior to publication. It is the responsibility of the author to ensure that no copyright infringement will occur by the publication of any material contained in their article.

EDITORIAL COMMENTS

The rest of this newsletter has been produced with ordinary spacing; this page will be presented in closely-typed format, so that you may decide for yourselves which you prefer. While this condensed form permits the inclusion of at least 30% more information per page, it reduces legibility by a considerable amount. It will be up to you to let me know which format you would rather have. If there is little or no response, I will produce all future newsletters in this condensed format, in order to allow as much information to be presented, and hang the legibility!

After sounding a few people out, I sent a circular to all those owners who are known to me in the Oxfordshire area. Of the 25 to whom I originally wrote, only 7 actually put pen to paper or picked up the phone and made a positive effort, for which I thank them. As I have decided to give the first two issues FREE to all 25 owners, this should give all of you a chance to sample the quality and decide whether you wish to continue receiving this newsletter. It is being produced on a shoestring budget (i.e., my salary) - I spent several days producing stencils and three hours one afternoon trying to make decent copies from them, but failed miserably; the idea was to make the newsletter available for just the cost of a second-class stamp each month. I have therefore decided to photocopy it, but for the moment in A4 format. Possible production in the A5 booklet format, with perhaps even reduction-copying, as was the case with TIHOME's sorely-missed Tidings, and as is the case with TI-99/4A Exchange's TI.MES, is a very strong possibility.

If you wish to be put on the mailing list, you MUST let me know and provide 12 second-class stamps to cover postage - with the special discount scheme currently operated by the GPO this could cost you less than £1.50 - and I have no real objection if you wish to copy the newsletter for distribution amongst fellow owners (you'll actually be saving me money!). Please note that I will also be providing a facility for the blind or partially-sighted, in that the newsletter will be read onto an audio cassette. Those owners wishing to obtain copies of the newsletter in this manner should provide at least one cassette (C60) and at least 12 'Articles For The Blind' labels. If there is sufficient demand for such a service from outside the Oxon group, I will seriously consider trying to provide it on a larger scale. Any assistance from interested members will be welcomed, especially tape-to-tape copying.

Make any contributions that you wish to: items for sale/exchange/wanted; large or small articles, programs etc.;comments; letters for publication; ANY queries which you may have (don't be backward about coming forward!) and if you wish you may remain anonymous - your wishes will be respected; and anything else which you think might interest anybody, but PLEASE: don't send large lists of your highest scores at Parsec or Invaders. It probably is very important to you, and if anyone wants to set up and run a regular section on Games Programming then by all means submit your efforts, but I hope that the newsletter will eventually gain some reputation as a good source of information both on the 99s and on Computing generally. Within the limits of the Copyright laws I will attempt to pass on any information which comes my way - undocumented functions on modules, system data, etc., etc.

OF THINGS TO COME

Articles planned for the next few months include: IAN SWALES' Improved Plotter and timings for functions in TI BASIC and Extended BASIC; DAVE HEWITT's suggestion for using your video recorder or TV modulator with your 99; 'Tokens' in TI BASIC, in Extended BASIC, and a few others from RICHARD BLANDEN; The Speech Synthesiser - How She Is Spoke; Obtaining non-keyboard characters in listings; additional CALLs which give you Enhanced BASIC; Sorting; filing; Using Your Joysticks; Assembly Language - the first steps; etc., etc. I look forward to hearing from you most of all.

BOOK REVIEW

LEARNING TO USE THE TI-99/4A by Kevin Townsend 110 pages £5.95 Gower Publishing

This book by Kevin Townsend is one of a series of eleven or more, all apparently written to a standard format which was developed by GARRY MARSHALL amongst others (you may have seen his recent book, GET MORE FROM THE TI-99/4A), and this approach turns out to be a mistake, as you will see later. The book itself is about A5 or so with about 110 pages. The cover price is £4.95, but for some reason everyone has to pay £5.95. The cover itself is also misleading, as it shows a VDU with what looks like a 64 x 30 display, which of course the 99/4A cannot achieve.

The foreword by the author sounds promising and the opening chapter is quite good, although I found it amusing that an illustration showing a floppy disk was in fact VISICALC for the Apple II. It did seem to suggest that the 4A could be regarded as a serious business machine, which of course it cannot, whatever the adverts might say. To make such a system out of the 4A you would need to expend far more time and money than on buying a dedicated system.

There then follows a historical section of questionable necessity, detailing the trials and tribulations of one Geophysical Services Inc., the forerunner of TI.

Getting down to specifics, the author makes the unforgivable error of saying that the 4A can control a cassette recorder. I take that to mean that it can select PLAY, RECORD, REWIND, FORWARD, or STOP, which it cannot; it can only perform the equivalent of a 'remote' PAUSE.

On page 9 it is stated that the 4A is supplied with a cassette cable. At the time that this book was written that was certainly NOT the case, and very few TI owners have been fortunate enough to be provided with that cable without the cost of it being added to the overall price of the console.

Also on page 9 is a photo showing a rear view of the computer with the legend "a number of the various peripheral connection sockets, including the cassette recorder port". So the power supply has now become a peripheral, and the TV socket must therefore be 'various'. Up to this point the only sensible recommendation I had come across was one regarding the disconnection of the remote lead when using the cassette recorder.

Things don't get better, however. The joystick port is called a 'remote control port', which to my mind means something to do with machine control - Robotics, maybe. The TV socket is called a 'display port', and the RS232/Expansion port becomes the Speech Synthesiser port.

In one illustration it looks as though the printer is connected via the cassette port, and although this possibility has been raised in Tidings, I don't think that it is a standard option.

The Peripheral Expansion Box gets a passing mention, which probably means that the author doesn't know too much about that either.

There then follows a sort of BASIC introduction, which is about as much good as a slap in the face with a proverbial wet kipper. The beginning is good (using PRINT and LET for openers as I had done in Tidings) but then it falls flat on its face by telling the reader that the symbol for exponentiation is the vertical arrow. It may be on some other machines, but on the 99s it is the caret (inverted V). This is followed by a chapter on strings and their manipulation, which has good exercises and has given me some ideas for my own books. However, the next section lets the side down again by stating that you can put your statement lines into the 4A in any order because the machine uses the line numbers to put them in the correct order. This is only half right: you CAN put the lines in in any order, but the computer stores them in the order in which they are entered, and then keeps a table to refer to their correct execution sequence (this information is courtesy of Stephen Shaw in Tidings).

Another boob is the inclusion of a program listing which gives PRINT "(CLS)", which might prove confusing to the newcomer. It does have another version which uses CALL CLEAR, but the first example shouldn't exist anyway. Another poor example is given when the book talks about being able to 'LIST from...to', and from my own experience it's a safe bet that more than one innocent reader will try to key that in verbatim.

Further evidence that Mr Townsend doesn't know his stuff comes when he says that you cannot add trailing spaces to an INPUT - you can if you use leading and trailing quotes ("). Maybe they thought that was too complex for beginners when proofing the book. At one point it even seems that there is an intention to make TI BASIC perform a statement of the type "IF N$ = G$ THEN PRINT N$", which of course it cannot.

There are many other silly things in this book which will trip the unwary, but what

is worse is the fact that half of the facilities are ignored, and more importantly, bad programming is encouraged. One example, if left running long enough, would eventually crash with a MEMORY FULL error. Elsewhere there is poor use of the most simple facility. For example, there is a five line program to effect the equivalent of CALL HCHAR(1, 1, n, 768). Someone has not been paying attention.

On top of this there is a lousy treatment of arrays, READ and DATA, and as usual a poor understanding of CALL JOYST(). Even the display comes in for a few errors:

'note that on a number of screens, only columns 2 to 28 can be accessed.'

Firstly, it's 3 to 30, and secondly they can all be 'accessed', it's just that on some badly-adjusted TVs you won't actually see the edge columns. The author also fails to distinguish between the screen area accessed by PRINT and INPUT, and that accessed by H/V/GCHAR().

The understanding of the Speech Synthesiser appears to be nil, and it is even called 'unique', which is bending the truth a little, and misses completely the fact that TI have offered a 'text-to-speech' facility on the Terminal Emulator II module for some time; a facility which is only now being considered by other micro manufacturers.

Then there are appendices. The first mentions the UK TI HOME Computer Users' Club (otherwise known as TIHOME and now extinct, but replaced by a group of exactly that name - I wonder how the author knew ?) and gives a run-down of two or three magazines, one of which I have never come across. There is a glossary running to some 8 pages, and one odd thing was the explanation of the jargon phrase 'WORD PROCESSOR'. I had not recalled seeing that mentioned in the main text, so I checked the index, and sure enough, there was a reference to it, so I checked the page. It referred to the glossary...

There is a listing for a game - Tower Of Hanoi - which some might find useful.

All in all I didn't find the book to be good value for money, nor do I think it a good buy for someone who doesn't already have quite a good grasp of the 99/4A anyway, in which case they wouldn't need to buy this book.

- -

BULLETIN BOARD

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

In this first issue we have a PRINTER for sale from someone outside the group. Gary Harding, whom some of you may remember as the author of articles on TMS9900 Assembly Language in TIHOME's Tidings, is selling his MICROLINE 82A printer to anyone offering over £200. It is an impact dot matrix printing at 120 characters per second (although a recent comparison in BYTE magazine with an EPSON FX-80, which is rated at 160 c.p.s., made it faster than the Epson), and I have seen the quality of its output - believe me, if I had the necessary folding stuff to hand I would buy it myself. It cannot reproduce the Texas graphics apparently, so this might make you think twice, although it is usually the case that you can access the dot printing facility via a short machine code routine.

You can contact Gary through me.

HELP HELP HELP HELP HELP HELP HELP HELP HELP HELP HELP HELP HELP HELP HELP HELP

A cry for help comes from Gary as well. He has bought a MICROLINE 92A and has produced his own cable to connect it to his 99/4A. However, he cannot get it to work. His 82A apparently had a serial and a parallel interface, and he used the serial port (RS232) for the past 15 months without difficulty. However, his 92A only has a parallel port, so he has had to use the PIO port on his RS232 card for the first time - and that's where he began to run into difficulties. He is not sure whether the fault lies in his cable or the PIO port on his RS232, and he asked me if anyone in the group could offer any assistance. He is willing, but not terribly keen, to entrust his card to the post so that someone could try the card in their own set-up and hopefully tell him which is at fault.

Again, contact Gary through me.

I have a limited amount of information on the 99s, some of which I am permitted to photocopy and pass on. Once I have assessed the cost of this (plus postage etc.), I will notify members through the bulletin board. I will also attempt to obtain permission from Texas Instruments and other interested parties for me to make a limited number of copies of other information for members' personal use only.

THE CONTROL AND FUNCTION KEYS

Peter Brooks (first published in TI.MES, newsletter of TI-99/4A Exchange)

The Users Reference Guide (URG) gives a passing and incomplete coverage of the CONTROL (CTRL) key, and a more detailed yet still incomplete explanation of the uses of the FUNCTION (FCTN) key. This article cannot do full justice to both keys, but it may begin to lift the mists which appear to surround the CTRL key especially.

To begin with, an understanding of the exact nature of ASCII is necessary. The American Standard Code for Information Interchange (the mnemonic is pronounced 'askey') is a system of characters and symbols in common use on computers. Each character or symbol has a code, just as in Morse code. Telegraphic Morse uses combinations of dots and dashes in varying numbers to represent letters of the alphabet, digits, and a few special combinations which represent standard words or phrases. Whereas one dot stands for 'E', three are used for 'S', so the groups of dots and dashes vary in size. To distinguish between groups, there is a slight pause after each group is transmitted.

ASCII uses groups of ones and zeroes, where the group size is fixed: 8 in all, a BYTE. The total number of possible combinations of ones and zeroes is 00000000 to 11111111 - i.e., 256. If you treat the combinations as BINARY numbers, then they range from 0 to 255 decimal:

128	64	32	16	8	4	2	1		
0	0	0	0	0	0	0	0	=	decimal 0
1	1	1	1	1	1	1	1	=	decimal 255

The full range of ASCII codes is only 0 to 127, not 0 to 255 as you might have concluded. Of the 8 BITS (BInary digiTS) in a byte, only 7 are used to represent all the letters of the alphabet (both upper and lower case - i.e., capitals and small letters), punctuation marks, the digits 0 to 9, the Arithmetic and Relational operators (+, -, /, =, etc.), and some special characters which are used to transmit instructions to certain items of equipment. The 8th bit is used as a check when transmitting character codes, called PARITY ERROR CHECKING. There are two methods of achieving a degree of validation of transmitted bytes; one is termed EVEN PARITY, the other ODD PARITY.

The 7 bits representing an ASCII character about to be transmitted are examined and the number of ones is counted. If the number of ones (1s) is ODD and the error checking method is EVEN PARITY, the 8th bit is set to '1' thus making the total number of 1s even. If the number is already even, then the 8th bit is left set to

zero. The opposite procedure is followed for ODD PARITY.

For example, take the upper case letter 'A'. Its ASCII code is decimal 65, or binary 01000001. There are two 1s, and using EVEN PARITY, the 8th or leftmost bit would still be 0. If however the method in use was ODD PARITY, the 8th bit would be set to 1, to make the total number of 1s odd, thus changing the combination to 11000001, which is equivalent to decimal 193 (128 + 64 + 1). This is where the codes 128 to 255 come in, for they are the codes 0 to 127 with the 8th bit set to 1. This is equivalent to adding 128 to the ASCII code.

Thus when receiving a transmitted byte of data, simple checks can be made to see if bits have been lost or changed - CORRUPTED. If the checking method is EVEN PARITY but a received byte has an odd number of 1s, then something has happened during transmission and the receiving equipment can either warn the transmitter that corruption has occurred, or ask for that particular byte to be re-transmitted until it is received in an acceptable form. Note that if the corruption was such that the total number of 1s was still even, then the method would fail to pick up the interference. Because the parity checks are not fool-proof, other checks are performed. One of these involves sending the data according to a PROTOCOL, where the receiving equipment is instructed to expect the data in a certain sequence. Any alteration in the sequence indicates corruption. For example, the start of a set of data might be indicated by one particular code or sequence of codes, followed by an indication of the number of bytes to follow - perhaps 64 every time - together with something called a HASH code. The ASCII codes of the 64 bytes to be sent are added together; each time the total exceeds 255, 256 is subtracted so that the final 'total' is always a number between 0 and 255. As each of the 64 data bytes is received by the equipment, it first scans the combination of 1s and 0s to make sure that there are an even (odd) number of 1s. If the byte passes this PARITY check, its value is added to a running total for that particular block - building up the hash code in exactly the same fashion as mentioned, subtracting 256 every time the total exceeds 255 - and the process continues until the 64th byte has been sent and has passed the PARITY check and been added to the growing hash total. The original hash code (which may have been transmitted either before or after the block to which it refers) is then compared with the hash code which has been calculated during transmission, and if they are both the same then it lessens the likelihood that any corruption has occurred.

Thus the transmission of the English word 'DEAF' will involve sending the codes 68, 69, 65, and 70. The binary equivalents of 69 and 70 contain odd numbers of 1s, so if we use EVEN PARITY we will have to set the 8th bits in those two bytes. This alters the codes to be sent: 68, 197, 65, and 198. The hash code for those four

is 16. If, by chance, during transmission the last code was to become corrupted - perhaps due to a sudden surge in the mains supply (mains spike) - and two 0s became 1s, making the 'F' become a 'P' say, the parity check would still pass the last byte. However, the ASCII code for 'P' is 80, and the calculated hash code would now be 154 and not 16. The hash codes disagree, therefore an error has occurred, and the computer can react accordingly.

On some computers, the ASCII codes 128 to 255 are used to make available so-called GRAPHICS characters. For complex reasons, TI's system only allows 96 to 159 on the 99/4 and 128 to 159 on the 99/4A (although 127 is 'blank' and could technically be regarded as the first of the graphics set), but at least the shapes of those characters are not predefined and fixed as they are on almost all other machines.

These ~~shape definitions~~ graphics characters have their shape descriptions (definitions) stored in a particular area of memory in the computer. Until CALL CHAR() is used to redefine those characters, that area of memory is used by the computer for its own purposes. One of these is the storage of an incoming program being loaded in TI BASIC using OLD CS1. This fact is used by a routine that I wrote a few years ago to monitor the initial process of OLDing, thus giving an early indication of a failure to begin loading. The routine simply prints all the User-definable characters (i.e., the Graphics characters discussed earlier) on the screen, and as the cassette OLDing begins, provided you have the volume level set correctly, you'll see the program arriving on board:

99/4

```
100 FOR I = 96 TO 159
110 PRINT CHR$(I);
120 NEXT I
```

99/4A

```
100 FOR I = 128 TO 159
110 PRINT CHR$(I);
120 NEXT I
```

Simply enter the relevant program and RUN it. When it has finished, type in the OLD CS1 instruction and follow the computer's prompts. When a program begins to load successfully, you will see an army of insects marching across the screen.

So far we have not even mentioned the CTRL or FCTN keys. As you will see next time, you could use the CTRL key to replace one of the programs above, placing the necessary characters directly on the screen from the keyboard. The FCTN key's effect is largely the same as the Shift key on the 99/4 - giving editing functions - except that there are certain aspects not covered in the URG. For example, pressing and holding down the FCTN key and then pressing 'V' will cause a character to be placed on the screen. Its ASCII code is 127 and it could be said to be the first of the graphics characters - the last character listed on page 102 of the URG - and its name is DEL; short for DELETE. Despite the fact that it is obtained via

the FCTN key, it is one of the CONTROL characters. The rest of these characters are obtained through the (wait for it) CTRL key. Control characters are used, as their name implies, to control things. On page 93 of the URG you will see the table of control key codes. Missing from ~~the~~ this list is CTRL , (comma), which has an ASCII code of 128 in BASIC mode, and 0 in Pascal mode. Its mnemonic is NUL. On some computers (e.g. most professional machines and the BBC, amongst others) these keys are 'active' - that is, if you press them, the computer will do something.

To confuse matters, the ASCII codes for these control characters are 0 to 31 and 127, but if you use the control key with the stated characters you will obtain on the screen characters with codes ranging from 128 (not forgetting NUL) to 159, which are the User-definable graphics characters. If however you were transmitting these codes down the telephone line, they would mean something to the receiving equipment. CTRL B would inform the receiver that you were about to send a section of text, and CTRL C would signal the end of that text. If the receiving equipment was a printer, the control characters would instruct the machine to perform a carriage return, for example, or to scroll the paper up by one line (line feed : LF), or even to select a different typeface.

The control functions were devised at a time when the Teletype was the main method of communication with the computer (under certain circumstances, the VDU or TV screen has been called a 'glass teletype'), hence functions like BEL, which causes a small bell to sound once, usually signifying the arrival of a transmission. On different machines the mnemonics may be slightly different, or have slightly different meanings (e.g. EOT can mean End Of Transmission, or End Of Tape - a reference to paper tape passing through a paper tape reader) but the principle remains the same.

Finally, to heap confusion upon the reader, there is an unintentional effect which can be produced using the FCTN and CTRL keys, which affords us a glimpse of the operation of the computer 'behind the scenes', as it were. Enter the following line but do not yet press ENTER:

```
1 REM
```

Now press and hold down the CTRL key, and then press 'A' so that both keys are being pressed - but don't hold them down for too long, or the auto-repeat function will come into play. Now press ENTER, and then instruct the computer to LIST the line. What is that following 'REM' ? According to the URG, CTRL A gives a character whose code is 129 - a graphics character; so where did 'ELSE' come from ?

Well, when you type out a normal line of BASIC commands (with a line number), the computer does quite a lot of work behind the scenes. It scans your BASIC commands,

checking each separate word against a list or dictionary which is already stored permanently in ROM (Read Only Memory). For each of the RESERVED WORDS (e.g. LET, PRINT, INPUT, CALL, IF, etc.,) the computer replaces that word with a TOKEN. This is a single character which, when the computer LISTs your program, will be translated back into the Reserved word which it is replacing. This means that although the computer will faithfully reproduce your listing when commanded to do so, internally it has used far less memory to store its version. This explains the phenomenon whereby you can enter a normal 4 line BASIC statement, call it up for editing, and go on to extend it up to 6 lines before encountering the LINE TOO LONG error. The token for ELSE is ASCII 129; for PRINT is 156, and so on. STEPHEN SHAW published a list of the tokens operative in Extended BASIC in an edition of Tidings, back-copies of which can still be obtained from PAUL DICKS, 157 Bishopsford Road, Morden, Surrey; tel: 01 640 7503 (after 8 p.m.). The issue in question, if you do not have it already, is V2.3 (1982). The list of those operative is too long to fit into the remaining pages of this article, and in fact the next issue will deal in some detail with the tokens, the further use of CTRL, FCTN, and editing to obtain characters on the screen as part of a listing (characters which cannot normally be obtained directly from the keyboard, but through POKEing - using MiniMemory, Editor/Assembler, or Extended BASIC + 32K) and their use to uncover additional CALLs which are present in the Personal Record Keeping and Statistics modules, CALLs which are not already known. I am indebted to RICHARD BLANDEN for initial information on these.

When you pressed CTRL A and entered the line, you placed the token directly into the program line, so that subsequent listing caused the Reserved word to be reproduced on screen. However, in order to do so, the computer had to go through quite a lot of additional processing, as you will discover if you experiment further with this. Instead of just CTRL A, if you had used CTRL A, B, C, D,... etc., and then ENTER and then listed the line, you would have had to wait quite some time for certain tokens to be translated, and for those tokens which exist, but which have no representation in TI BASIC, the wait can amount to several seconds or more. If you edit the line by calling it up but not changing it in any way and then press ENTER, it goes back into memory in exactly the same form. If however, you edit the line by pressing a key but do not change the line - that is, if the cursor lies over the 'R' in 'REM' and you press 'R', thus editing without changing - the computer will assume that you <u>have</u> made a change, and will scan the line(s), taking the words as they appear on screen rather than their tokens, making subsequent listing much quicker. There is far more to this than at first meets the eye!

These are all examples of the multitude of ways in which the value represented by the code of a character called up by FCTN, CTRL, or other keys, can be interpreted, depending upon the circumstance in which it is used. The ASCII character whose code

is 159 can be regarded as a graphics character (CTRL 9), or character 31 with the most significant bit set to 1 (cf PARITY), or the token for the Reserved word OPEN, or a small number (represented one of two ways: 'unsigned', where the value lies in the range 0 to 255 - i.e., 159; or 'signed', where the value lies in the range -128 to +127 and the most significant bit (leftmost or 8th bit) indicates whether the value is positive or negative - i.e., -97. I will try to fit the discussion of ONES and TWOS COMPLEMENT into a later issue), or the shape definition of a line of eight dots horizontally in a graphics character (in binary: 10011111; in hexadecimal: 9F), or even as part of a TMS9900 machine code instruction or datum.

- -

'TEACH YOUR GRANDMOTHER TO SUCK EGGS' DEPARTMENT

A former member of TIHOME, one DAVID BROWN of Abingdon, borrowed my SPEECH EDITOR module manual and discovered that the SPEECH SEPARATOR CHARACTERS detailed in that manual are also active in Extended BASIC, and ROBERT BATTS of TI (who will be reading this newsletter) tells me that details of the use of the hash symbol on either side of a 'phrase' were also omitted from the Extended BASIC manual. I don't know who else in the group has a Speech Synthesiser and doesn't know about these items, but anyone who is interested can obtain further details from me (see elsewhere). As an example, the '+' symbol reduces the pause between the enunciation of two words or phrases to almost nothing, so that CALL SAY("A+B") will be faster than just CALL SAY("A","","B").

Following this surprise (I've had my Speech Editor for more than 3 years and my Extended BASIC for more than two and I hadn't noticed anything amiss), a member of TI-99/4A Exchange wrote to me through its organiser, CLIVE SCALLY, and discussed the CALL KEY() bug, which reared its ugly head some time ago when I wrote an article in Tidings on High Resolution Plotting (all 14,000 words of it in one issue alone!). One of the routines provided cursor control of the plotting, using CALL KEY() with key units 1 and 2. I soon found out from those trying it on their 99/4As (I'd prepared everything on my trusty old NTSC 99/4) that using the X key with key unit 1 didn't work as it should. If you write a short routine to scan the left hand half of the keyboard using CALL KEY(1, K, S) and then print out the value of K whenever a key is pressed, you will find that pressing 'X' results in what looks like a 0. If you include a relational expression (PRINT K = 0) and press 'X', you will get the value corresponding to FALSE - in other words, although the screen says it's 0, the computer says it's not. If you print EXP(LOG(K)) when pressing 'X', this time the computer tells you that the value is really 400! The same thing happens on the scan of the right hand half of the keyboard using key unit 2. Stephen Shaw

suggested using a test for (K + 1) = 1 in the plotting routines, which did work, and I couldn't think of anything better, so that is what we advised people to use. Recently SIMON PRYCE wrote via Clive and suggested using a simpler relational expression: K< 1 which works fine, and occupies less memory and is faster in execution. If anyone in the Oxon area has NOT experienced any difficulty with a CALL KEY() bug as detailed here, perhaps they would like to get in touch with me, and let us see if we can discover exactly what the difference(s) might be between their machine and mine. Perhaps we could shed a little light on the mystery.

CLOSE FILE

Well, that is the first issue of the newsletter. I apologise for the scrappy layout, (and possibly appearance - I haven't photocopied it yet!), but with your help, criticisms, etc., I hope that it will go on to grow and improve and become a useful source of reference material for the TI programmer and user. Don't forget to make the effort and respond with ANY comments (naturally, praise will not be refused), and PLEASE don't forget: the first TWO issues are FREE; thereafter you will only get a copy if you have responded either by phone or post and sent in 12 second-class postage stamps to help out with my postage costs.

Oh, and before I forget: there are NO April Fool items in this issue. Maybe next year, when we're running at 2000 copies a month...

If sufficient members live close enough to be able to hold regular meetings (or any meetings at all) amongst themselves, would they like to broadcast their efforts through these pages ? You will be more than welcome - we may end up with several subgroups - and any support which you think I may be able to give, please do not hesitate to ask.

I would like to put members with similar interests in touch with each other, and in the next issue there will (hopefully) be a simple questionnaire which you may fill in if you have no objection, and return to me in complete confidence.

I would also like to run a page or six for the younger members of the group, to be written, edited by them. Do we have any takers ? Let me know as soon as you can.

Good programming,

Pete Brooks