INTERNATIONAL TI-LINES

Volume 4, Issue 2 July 1st., 1987

Formerly OXON TI USERS

A PUBLICATION OF THE INTERNATIONAL TI USER GROUP

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

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

OXFORD 510822

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

READ THIS FIRST !.....3
A (short) editorial by another name

MACHINE CODE.....5
A brief summary of Source and Object code differences

LETTERS.....10
JOHN ROE and DEREK ALLEN

MATHEMATICAL PROBLEMS YOU CAN SOLVE WITH YOUR TI-99/4A.....13
FRANCESCO LAMA on solving equations

PROGRAMS THAT WRITE PROGRAMS.....16
JIM PETERSON's thought-provoking series continues

THE PRINTER'S APPRENTICE.....19
STEPHEN SHAW supplies an example of mixed text and graphics

VOLUME 1 : ARTICLE INDEX.....20
Not a full index, just a list of the articles which appeared in 1984 - 1985...

VOLUME 2 : ARTICLE INDEX.....25
And in 1985 - 1986...

NOTICE BOARD.....29
Good Sales, Good Buys

COMMERCIAL SOFTWARE/FIRMWARE LIST.....30
The latest update to the list, with 16 new entries and some price reductions

THE TIHOME SOFTWARE COLLECTION PRICE LIST.....32
The full price list for the Collection

!!! Apologies for the use of the £ instead of the hash symbol in this issue; lack of time has meant a rushed production and inevitable glitches. PB

Coming soon: BILL REED's RS232 and CENTRONICS interfaces for use with MiniMemory. MIKE GODDARD's DIY Expansion System. Reviews of the GRAM KARTE from MECHATRONICS. A tale of woe with a happier ending from EDDY TURNER. More on machine code from yours truly, and much more.

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

Would you believe that TI-LINES is 40 issues old this month ? A large quantity of water has flowed under the bridge in the intervening period (and large quantities of hot air have been released at times) but I hope that occasionally material from IT has found its mark and solved a few readers' problems. If I'm REALLY lucky this month, I will have been able to improve the typeface used throughout the magazine, but knowing Brooks' Second Law (Never On Time) I wouldn't count on it!

A couple of things were omitted from the last issue: firstly, the small character on the cover was taken from a letter sent in by STEPHEN SHAW, demonstrating the versatility of a new piece of software.

Secondly, the sorting routine didn't say anywhere that the file (once you have typed it in!) should be placed on a disk with an appropriate file name. In the example given, the file was called SORTO, which follows the convention whereby OBJECT code files usually have an O as a terminating character, while SOURCE code files have an S.

At the time of writing you might like to know that resubscription levels are high - we number 160 with 80 left to resubscribe, and we are still taking on new subscribers.

This influx of new members is due to an advert placed in MicroComputer Mart, a permanent entry on page 703 of CEEFAX, recent ACC editorial in PERSONAL COMPUTER WORLD, a letter published in POPULAR COMPUTING WEEKLY by NEVILLE BOSWORTH (good on yer, Neville), and word of mouth. Much of the word of mouth has come through GORDON PITT of the West Midlands TI User Group, and to many other ITUGers who have been putting themselves about recently. To all, grateful thanks. If the gods smile upon us, we may yet grow larger and stronger as the months go by.

I don't think that the month of June could have been more eventful if it had tried. What with the new subscriptions, new interest in the TI, the deletion from CEEFAX of the ITUG entry for some as yet unexplained reason, one of my sisters is, at the time of writing, waiting to deliver her second kiddie, and a close friend of mine for some 23 years has just had a heart and lung transplant. That's in addition to minor events like the general election...

If, therefore, you have been trying to contact me by phone and only been able to get the answering machine, well, now you know why. My apologies to one and all, but I can't be in two places at once and often I have been late in getting back home. Normality may not be restored for some weeks to come, I'm afraid.

I am taking my life in my hands this month and trying to write an intro article for those aspiring to become Machine Code Beginners (or should that be aspirin ?). However, after the first 20 pages without having

))))))

got any closer to describing 9900 code I began to wonder just what I had let myself in for... I have held back for a month or so from actually committing myself to publication in case I find a simpler approach!

In its stead, I have produced an article which briefly gives details about things like SOURCE code and OBJECT code, and what the differences are. I hope that it helps some of you out.

As time passes, so this editorial is updated. Thus it is that I can happily report our re-instatement on page 703 of CEEFAX. Wouldn't Life be boring without all these exciting little moments to fill it ? Don't answer that!

I get some odd items sent to me from time to time. The most recent, and worthy of mention, came from JACK SHATTUCK of the USA, and consisted of a disk containing a write-up of the American Surgeon-General's report on AIDS (Acquired Immuno-Deficiency Syndrome). In the course of my daily (as opposed to every-other-minute-of-the-daily) work I wear several caps and one of them is as a Safety Officer in the hospital laboratory where I am a medical laboratory scientific officer (among other things!).

Because my research lab has handled "unscreened" blood and other samples in the recent past, it is categorised as being "at risk" as far as not only AIDS (strictly, HIV infection) but also a range of other nasties are concerned. This automatically means that I have a vested interest in keeping an eye on the AIDS research, and that I have received a more detailed breakdown on AIDS than that given to the general public.

By and large, the information dished out on TV and in the papers has been "politic" - that is, expedient, rather than strictly accurate. The old political adage of never admitting you don't know about something and haven't got an answer still holds true unfortunately, and it has occasionally bugged me to see TV "personalities" making categorical statements about the nature of HIV and its infectivity.

However, the waters have already been muddied many times and it would serve no real purpose to air the subject in TI-LINES, quite apart from the fact that most readers will either be sick and tired of hearing about AIDS, or be too apprehensive to read anything further.

Therefore, if anyone with a disk system wishes to receive a copy of the Surgeon-General's report (single sided, single density, 40 track - but could be 35 track if you want it) then please send me a blank disk plus sufficient packing and return postage and I will copy the disk I have for you. I am willing, if pressed, to provide a printout of the entire document for a small fee, but this will require some time to implement and time is one commodity I don't have in abundance at present!

Quite frequently recently I have seen adverts for disk drive controllers for sale, with the Disk Manager module being offered as a separate item. This is rather naughty - it is like selling a car to one motorist, and the keys to another. The two go together, or they ought to...

))))))

M A C H I N E C O D E

A few enquiries have mounted up over the last few months to the level where a brief description of the nature of, and differences between, source and object code is warranted.

SOURCE CODE

This is usually a DISPLAY, VARIABLE 80 (DV80) disk file, created using either the Editor from the Editor/Assembler (EA); the Editor from the TI-Writer/Funnelweb package (FWEB) - saved to disk using PF, not SF, to eliminate the terminal (tabs) record; or a homebrewed editor.

It consists of the instructions to the computer written as MNEMONICS; i.e., LWPI 1234, ADD R1,R2, and so on. Specific points in the code are identified not by line numbers but by LABELS:

```
CHANGE MOV   R5,R5
       JNE   SMALL
LARGE  INC   R5
       LI    R0,>01E1
       BLWP  @VWTR
       JMP   LOOP
SMALL  CLR   R5
       LI    R0,>01E0
       BLWP  @VWTR
       JMP   LOOP
       END
```

In this example, the labels are CHANGE, LARGE, SMALL, and one which does not appear here, LOOP.

This is often called ASSEMBLY LANGUAGE. Technically, Assembly Language is the instruction set associated with an ASSEMBLER, which is the program whose task it is to translate the SOURCE CODE (i.e. the near-English instructions above) into OBJECT CODE. There are at least three recognisably different forms of object code used on the 4A (see later).

The Assembler converts the instructions contained in the Source file into a form suitable for loading into the machine under differing circumstances (see later). It expects the instructions to be presented in a particular way, according to a specific format.

Just as TI BASIC has RESEQUENCE, LIST, EDIT, etc., which are not BASIC instructions but commands for the BASIC Editor to obey, so the Assembler likewise has a small set of similar instructions - called ASSEMBLER DIRECTIVES. Examples of these are DEF, REF, EQU, DATA, BSS, END, AORG, RORG, and so on.

>>>

When the Assembler program is run, it takes a source code file as its input, and outputs an OBJECT CODE file according to the instructions given by the User.

A convention exists whereby the names of the files indicate their type.
A terminating S, as in CHAR/S, CHAR_S, CHARS, indicates source code.
A terminating O indicates usually UNCOMPRESSED TAGGED OBJECT CODE - e.g. CHAR/O, CHAR_O, CHARO. COMPRESSED TAGGED OBJECT CODE is often indicated by a trailing C - CHAR/C, CHAR_C, CHARC.

PLEASE NOTE: filenames which end in this way are NOT automatically machine code.

The terminators _S and _C are used, for example, by some graphics programs to indicate a SLIDE file or a COLOUR file to the program, so only use the filename format as a GUIDE when dealing with unknown files.

If a file is named GAMES, and it is DV80, it may possibly be the Source code for a machine code game, or it may be a text file discussing the subject of Games. Anything DV80 or DF80 (Display, Fixed 80) can be loaded into FWEB or EA, or you can write a simple Extended BASIC (XB) routine to read the file, so examine it and find out what it is before making any assumptions!

OBJECT CODE

Object code is produced from Source code by an Assembler. There are three formats for the 4A buff to get thoroughly confused over.
UNCOMPRESSED TAGGED OBJECT CODE, COMPRESSED TAGGED OBJECT CODE, and UNTAGGED or PROGRAM FORMAT or MEMORY IMAGE (most commonly used name is PROGRAM FORMAT).

What's a TAG ? A tag is a character which provides information to a LOADER. What's a LOADER ? A loader is a routine which is used to transfer object code from an external storage device (e.g. disk) into the computer's memory. At least three loaders are used on the TI (as far as machine code goes) and all operate slightly differently. They each expect the Object code to be in a particular format. Any errors or deviation from the expected format, and the loader may not operate.

A Tag is like a DIRECTIVE. There are numerous tags which are available to the loaders, and they consist of single characters.

They indicate to a loader either the nature of the data (object code) immediately following the tag, or the location to which subsequent data should be sent, and so on.

The XB/EA tags are:

0	module ID	B	absolute data
1,2	entry address	C	relocatable data
3,4	external references	D	load bias
5,6	external definitions	E	undefined
7	record checksum	F	end of record
8	ignored checksum	G,H	undefined
9	absolute load address	I	program segment ID
A	relocatable load address	:	end of file

››››››

Uncompressed tagged object code consists usually of "fields" of groups of hex digits. Usually a tag is followed by four hex digits at a time (depending on the tag) which represent two ASCII codes.

```
A0012B0300B0000B020AB1000BC80AB83CCBF820C0000B83FDBD820C00007F2C6F
 ^    ^    ^    ^    ^    ^    ^    ^    ^    ^    ^    ^    ^    ^
```

The tags have been indicated by "^", which of course would not normally be present.

If you have issue V4.1 to hand, look at page 9. The machine code there is a practical example of uncompressed tagged object code.

The tagged object code files are usually DF80 format, and may be examined/edited using EA/FWEB/homebrewed utility. Editing using FWEB must be followed not by SF (for SaveFile, which uses DV80 format) but by PF (for PrintFile) with the F option (to print out in DF80).

The groups of four hex digits are converted by the loader into two ASCII coded characters and stored onboard in memory. Something like hex 5A5B is actually 5*16+10, and 5*16+11, or characters 90, and 91. An actual example is 0420 (look out for it on page 9 of V4.1) which becomes CHR$ 4 and CHR$ 32, and which is the Object code equivalent of the Source code instruction BLWP.

At the end of the uncompressed tagged object code file is a list of the name(s) or label(s) of the routine(s) which are to be accessed using for example CALL LINK from Extended BASIC. An example from V4.1 is the segment consisting of 50AF6SORT 7FD1FF. SORT is the name of the routine.

Compressed tagged object code is slightly different. The data to be loaded into memory is still preceded by tags, but this time the format is not four hex digits per field, but the two ASCII characters themselves. Here is an example taken from the LDFW file associated with FWEB:

```
 BAXB RBOBBUSBTUBS BneBedBs BtoB kBnoBw BwhBicBh BdiBskB dBriBveB cBonF
 BinBs B tBheB FBUNBNEBLWBEBB sBysBteBm BfiBleBs.B DBSKB?TBryB aBgaBinF
 BitBh BriBghBt BdiBskB !B  B  B  B  B  B  B  B  B  B  B  B  B  B  B  F
```

This is actually a segment chopped out of the middle of the file, but it contains sufficient "legible" characters to be useful. I use the term "legible" because each character can have an ASCII code ranging from 0 to 255 decimal, and only a few of those are printable here. This is a limitation imposed by my printer - in FWEB, some of the characters whose codes are 0 to 31 have had their shapes defined so that they represent the hex equivalent of the ASCII code. See Jo Ann Copeland's TI-Writer tutorial series for a discussion of CTRL U mode which allows the full use of these characters.

The use of a compressed format means that a file is only half the size of an uncompressed file, and this in turn means that such files occupy less space on disk and load more rapidly. However, it is easier to edit an uncompressed tagged object code file (puff, puff) since you are

))))))

only ever dealing with the hex digits 0 to F. It is not so easy to examine such files, since you need to convert from hex to decimal in order to recognise what you are looking at (unless you are already adept at converting in your head).

In an ideal world, any editing will be performed on the Source code file and the result re-Assembled. We do not live in an ideal world, and so Source code files rarely reflect accurately the Object code files they once produced.

Program format files are tag-free and basically consist of a straight copy of a block of memory. If you were to load either a compressed or an uncompressed tagged object file into RAM, and then save a copy of that block of RAM out to disk, you would basically have a Program Format file. (I make no claims as to the absolute accuracy of this!).

As such, program format files occupy the least amount of space on disk, are the fastest to load from disk into memory, and are the meanest to try to edit. Why aren't all machine code files stored and used in this way if it is so fantastic ? I don't know - probably because it requires some additional work and programmers are supposed to be inherently lazy.

POSTSCRIPT

There are at least two other methods of getting machine code (object code) onboard (neglecting any routines you may have developed for your own use).

1) Poke it in byte by byte. Good examples of this approach are JIM STRINGFELLOW's programs in the recent past (published in IT).

   There are two main methods:

   a) DATA statements in an appropriate BASIC program (XB, MM, EA)
   b) Separate data file(s) on disk/cassette/other

2) "Hidden" or "masked" code. This is a weak description of code which is brought onboard in the guise of being an Extended BASIC program, where the actual XB routine simply inserts the required entries in the REF/DEF table (routine labels and start addresses), using the poke-it-in approach.

   This technique is used in FWEB and probably numerous others. How can you tell ? Clue 1: the XB program is only a few lines long (often no more than two or three) yet occupies an enormous amount of space on disk/cassette - many times its "true" size. What is perhaps 1 or 2K of XB may "hide" 12K or more of machine code.

   Clue 2: the XB program will CALL INIT and then CALL LOAD() a series of values into memory (poking them in byte by byte). If you convert the values (usually numbers in the range 0 to 255) into ASCII characters, you may be able to spot the routine name(s) at least. Finally, a CALL LINK() will invoke the operation of the machine code routine(s).

))))))

Why is the "hidden" format used ? I am told that the XB loader used to bring XB programs on board is faster in operation than the program format loader and much faster than the usual CALL LOAD(filename) loader.

Different Environments

There are several different loaders in use on the 4A, most being pretty exclusive (which means that they'll only load files which follow their own particular format). XB has two, EA has three, MiniMemory (MM) has I think 2. Some loaders appear compatible on the surface, but are incompatible in detail and cannot load each others files.

Hints on recognising EA program format files:

A limitation placed on EA Program format files is that the utility used to create them can only work with 8K blocks at a time, so any file which is greater than 8K is split up into separate 8K (32 sector) files, each named slightly differently. The last character of the first 8K block filename is incremented for each subsequent 8K block filename. The program format loader expects them to be stored in this way.

If you therefore encounter two or more program format files on disk, whose filenames are identical in all but the last letter, and if the last letter in each case is in sequence - e.g. DISK, DISL, DISM, or TENNIS, TENNIT, TENNIU, then you can be reasonably sure that you are dealing with a group of files which are all machine code and all loadable with the EA "option 5" loader, or equivalent FWEB loader.

Note that the above constitutes a "brief" summary description; wait till you try wading through the EA manual!

If you have any further specific queries which have been raised by this article, either by my incompetence in explaining something or in having omitted something vital, please drop me a line. At the time of writing, my workload is such that personal replies are being delayed by up to 6 weeks (a situation which will hopefully not endure!) PB.

))))))

L E T T E R S

I received the following letter from JOHN ROE concerning shuffling, a subject which has cropped up a couple of times recently:

"Dear Peter,

I enclose a program which I have been using for nearly four years now, and which uses a shuffling routine similar to that in V4.1 TI-LINES. As you will see, the routine at line 280 is formally identical to lines 150 to 200 of your routine. I had meant to write to you about this before now, but I had mislaid issue V3.12 shortly after receipt and have only just found it again.

The program is designed to pair off players (or teams) in a knockout competition, and follows the usual convention that all byes (if any) should appear in the first round only, leaving the remaining rounds to number only successively decreasing powers of two.

Lines 230 to 260 set up the screen, initialise variables and set up the array with players' names. Lines 270 to 300 do the shuffling and add a bit of 'User friendliness'. Lines 310 to 350 calculate the number of byes required. The byes are then displayed separately followed by the pairs. Lastly, lines 360 to 450 provide an option to produce any required number of printouts.

I must say that I did not find the basic shuffling routine particularly difficult to work out; I had more difficulty with the routine to calculate the required number of byes so that the next and succeeding rounds would number only an exact power of two.

Just a point about your remarks in paragraph 3 of page 5 of V4.1. Count me definitely as one (at least) of the 'few'. I do know how to spell, and can write good grammatical English. I must admit however that both spelling and grammar leave much to be desired among computer buffs in general. From a magazine I once entered a program called SENTINAL - the programmer's spelling, not mine. The spelling error may seem of little real importance, but as it happens the title was designed to be displayed as quadruple size characters - that is, double height and double width.

Each letter therefore required four character definitions and if you spell the title correctly (as SENTINEL) you don't need the definitions to construct the letter A, thus saving time in setting up the title. I also managed to alter the program so that this setting up occurred while another earlier screen was being displayed. "

{John's program is listed overleaf and is designed to run in Extended BASIC. If anyone else has something to contribute concerning the process of shuffling, the subject is still open. PB}

)))))))

```
 100 REM ---KNOCKOUT DRAW---
 110 REM ----------BY---------
 120 REM -----J. T. ROE-----
 130 REM --------------------
 140 REM -ALTER: DIM A$(40)-
 150 REM ----IN LINE 240----
 160 REM TO TAKE THE MAXIMUM
 170 REM -NUMBER OF PLAYERS-
 180 REM --LIKELY TO ENTER--
 190 REM --------------------
 200 REM ALSO ALTER LINE 400
 210 REM TO FIT YOUR PRINTER
 220 REM --------------------
 230 CALL CLEAR :: CALL SCREEN(6):: FOR I=0 TO 14 :: CALL COLOR(I,16,
 1):: NEXT I
 240 OPTION BASE 1 :: DIM A$(40):: D=2 :: E=0 :: RANDOMIZE
 250 DISPLAY AT(8,2):"HOW MANY PLAYERS? " :: ACCEPT AT(8,22)BEEP
 VALIDATE(DIGIT)SIZE(3):N
 260 FOR I=1 TO N :: DISPLAY AT(8,2)ERASE ALL:"ENTER PLAYER No. ";I
 :: ACCEPT AT(10,2)BEEP VALIDATE(UALPHA,".")SIZE(26):A$(I):: NEXT I
 270 DISPLAY AT(10,5)ERASE ALL:"PLEASE WAIT.....": : :"        SORTIN
 G IN PROGRESS"
 280 FOR I=N TO 1 STEP -1 :: R=INT(RND*I)+1 :: T$=A$(R):: A$(R)=A$(I)
 :: A$(I)=T$:: NEXT I
 290 DISPLAY AT(10,5)ERASE ALL:"SORTING COMPLETED" :: CALL WAIT(200)
 :: DISPLAY AT(10,1)ERASE ALL:"PRESS ANY KEY TO SCROLL LIST"
 300 CALL WAIT(300):: DISPLAY ERASE ALL
 310 D=2*D :: IF D<N THEN 310 ELSE IF D=N THEN 340
 320 E=D-N :: FOR I=1 TO E :: PRINT A$(I);TAB(18);"BYE": :
 330 CALL PRESS :: NEXT I :: PRINT : : :
 340 FOR I=E+1 TO N-1 STEP 2 :: PRINT TAB(5);A$(I): :TAB(10);"V": :
 TAB(5);A$(I+1): : :
 350 CALL PRESS :: NEXT I
 360 DISPLAY AT(10,1)ERASE ALL:"PRINT PAIRINGS? Y/N"
 370 CALL KEY(3,K,S):: IF K<>78 AND K<>89 THEN 370
 380 IF K=78 THEN CALL GOODBYE :: STOP
 390 DISPLAY AT(10,2)ERASE ALL:"HOW MANY COPIES? " :: ACCEPT AT(10,20
 )BEEP VALIDATE(DIGIT)SIZE(2):C
 400 OPEN #1:"RS232.BA=1200.TW.LF.DA=8.PA=N",FIXED 80
 410 FOR J=1 TO C :: IF D=N THEN 430
 420 FOR I=1 TO E :: PRINT #1:A$(I);TAB(22);"Bye" :: NEXT I
 430 FOR I=E+1 TO N-1 STEP 2 :: PRINT #1:A$(I);TAB(19);"v    ";A$(I+1)
 :: NEXT I
 440 PRINT #1:" " :: PRINT #1:" "
 450 NEXT J :: CLOSE #1 :: CALL GOODBYE :: STOP
 460 SUB WAIT(D):: FOR I=1 TO D :: NEXT I :: SUBEND
 470 SUB GOODBYE :: DISPLAY AT(10,6)ERASE ALL:"GOODBYE" :: CALL WAIT
 (300):: SUBEND
 480 SUB PRESS
 490 CALL KEY(3,K,S):: IF S=0 THEN 490
 500 SUBEND
```

NOTE: When you see a £ in the listings, it is a #

))))))

DEREK ALLEN also sent me a letter expressing his views about TI-LINES and offering assistance (as have others, for which many thanks, and I will pursue some of the more practical avenues):

"Dear Peter

Having read the June editorial I felt a letter was in order.

It is the way of things that people are quick to complain and slow to praise and support. It can extremely difficult to present an article in a way which holds the interest of everybody throughout, and even more so to do the same for an entire magazine.

I have just submitted a small article of a non-technical nature to EAR 99ers (SCOTT & JO ANN COPELAND). Compared to most, it is short and very simple, but still took me several evenings to complete. Being my first attempt at "serious" writing, I could not believe how difficult it was to get balance and flow into it.

I now appreciate the work that goes into producing such things, and I think that you do wonderfully well to achieve what you do.

I have returned my questionnaire but perhaps a few qualifying remarks may be appropriate.

Editorial is absolutely essential, it ties everything together. It is impossible to please all of the people all of the time, the more so in our field since there are so many facets to Computing.

My system is rather basic (just Extended BASIC and Speech) and I am definitely no boffin, but I still read all articles even if I cannot fully understand or implement them because one can ALWAYS learn something.

My TI-LINES' are not just a monthly read but a reference library which is kept in order on my shelf and used as such.

An index would obviously be an excellent idea, so good luck to Peter Killick.

The actual quality of the magazines - printing, etc. - is quite adequate and if it is legible what more is needed ?

I am sorry if it appears that I do not have any constructive criticisms for you, but the magazine seems OK to me.

As I have said, I am no boffin and cannot write technical articles. My main interests are Adventuring and fooling around with Maths and Speech. I would love to be able to afford a printer so as to get into Word Processing etc., (so as to spare people my awful handwriting!).

If you have any ideas on how I might be of help to you, I am willing to give them a bash.

Meanwhile, could we have some more details on the ROMOX cartridges ? Do I understand correctly that we can have ANY program we like on them ?"

{At present we are limited to certain software. When a list is finally available I will publish it. PB}
}}}

MATHEMATICAL PROBLEMS YOU CAN SOLVE WITH YOUR TI-99/4A

By FRANCESCO LAMA

I think we all must remember saying "I can't solve that equation!" at some time during our school days. On some occasions a lot might have depended on our being able to do so (like during exams!) but our memory failed us... We couldn't think of a way. More often than not we would have been given a problem such as:

$$12X^2 + 3X + 8 = 0$$

which simply means "Find the values of X for which this is true", and we either couldn't call to mind the formula which gives us the answer, or our memory half-failed us and we only partially-remembered the correct formula.

However, one way or the other we managed to survive exams and a few masochists like me even decided to take up solving complex mathematical problems as a profession.

What I am going to describe here is a pair of simple logical algorithms which one is forced to use when the equations one is trying to solve are "transcendental" - a word on which to meditate but which basically means "too difficult to find a formula which will give the answer".

I must also add that I claim no originality for the methods, which have been known for hundreds of years (back to times when there were no fast computational ways of implementing them), and are now to be found in the Maths libraries of most pocket calculators.

The first concept one needs to introduce is that of a function of a single variable "X". This can be approximately done by saying that F(X) equals any combination of the functions present in the computer Maths library involving any number of constants but only one variable - X.

For example:

```
F = SQR(123*(ATN(4.24*X)))/.3695*X+LN(254.1*SIN(4*SQR(34.78*X)))
```

The second concept which is needed for the introduction of the methods below is the one of the "continuity of the function" - i.e., for all practical purposes nothing "funny" should happen to it when we try to calculate it (like LN() of zero or of a negative number, division by zero, SQR() of a negative number, etc.) - in other words, the function F should be "well behaved" in the region of the values of X for which we are trying to evaluate it.

One of the most common problems in both Physics and Mathematics is that of finding when a function is equal to a certain number. For example, finding when F = 4.532, which is equivalent to determining when

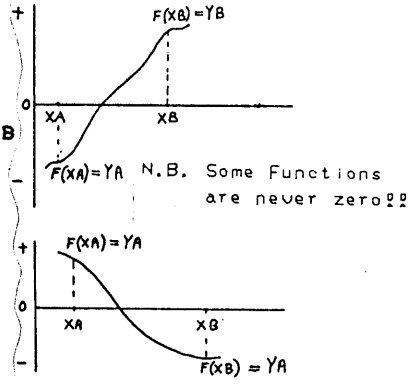
))))))

F - 4.532 = 0. This reduces the problem to finding when a certain function equals zero (does it begin to sound like the questions you were set at school ?).

So far I have been trying to debunk the complexity of Mathematics and it may all have been rather boring, but I am sure you will find the following - er - even more boring!

The logical procedure to find the value of X for which a function F is zero relies on the definition of an interval of X values (given by its two extremes XA and XB, either of them can be the larger) along which F changes sign - i.e., goes from a negative value at XA to a positive value at XB or vice versa. Therefore lines 90 to 170 of the program given below were written in order to ensure one of the two situations shown graphically beside the program lines:

```
 90 CALL CLEAR
100 INPUT "DEFINE INTERVAL XA,XB:":XA,XB
110 X=XB
120 GOSUB 1000
130 YB=F
140 X=XA
150 GOSUB 1000
160 YA=F
170 IF YA*YB>0 THEN 100
```



Otherwise, the program simply requires you to enter a new pair of limits!

The next section of the algorithm can take one of two forms: the first is called the Bisection method, and the other is called Newton's method, the two differing simply in the way they estimate where the zero of the function is likely to be, given the two extremes XA and XB. The former simply calculates the value of F in the middle of the interval XA-XB (i.e., at the point (XA+XB)/2) and if it finds the function to have the same sign as at the point XA it assigns the value of the midpoint to XA and otherwise to XB (remember the function has opposite signs at XA and XB).

The general aim of this procedure is to narrow down the interval over which a zero is to be found, still maintaining XA on one side of zero and XB on the other, until the absolute value of the difference between XA and XB is less than the REQUIRED PRECISION (with which you want to know the result), and which you are asked to enter at line 172 (see later).

On the other hand, Newton's method chooses to approximate F with a straight line between the points XA and XB and finds the point where this straight line is zero. For this purpose it not only has to

>>>

remember XA and XB but also the values of F at XA and XB (in the program they are called YA and YB). It then finds the point X where this straight line crosses zero and takes it as the new intermediate point in the interval XA-XB (directly replacing the midpoint for all the purposes of the program).

The two programs below differ only in line 410 of the subroutine at 400; because of the different choices of intermediate point and in lines 180 and 210 of the main program because of the necessity for Newton's method to remember the values YA=F(XA) and YB=F(XB).

The latter method is faster at converging to the required accuracy and both are widely used in scientific numerical computations. Are you still awake ? Sorry if I'm boring you to death!

The remaining lines, written in Extended BASIC, are as follows:

1)

```
172 INPUT "REQUIRED PRECISION: "
    :E
175 GOSUB 400
180 IF Y*YOLD>0 THEN XA=X ::
    GOTO 175
190 XB=X
200 GOSUB 400
210 IF Y*YOLD>0 THEN XB=X ::
    GOTO 200
220 XA=X
230 GOTO 175
240 PRINT X;Y
250 STOP
400 YOLD=F
410 X=(XA+XB)/2
420 GOSUB 1000
430 Y=F
440 IF ABS(XB-XA)<=E THEN 240
450 RETURN
1000 F=((X+1)^10-1)/X/(1+X)^10-
     7.889
1010 RETURN
```

2)

```
172 INPUT "REQUIRED PRECISION: "
    :E
175 GOSUB 400
180 IF Y*YOLD>0 THEN XA=X :: YA=Y ::
    GOTO 175
190 XB=X
200 GOSUB 400
210 IF Y*YOLD>0 THEN XB=X :: YB=Y ::
    GOTO 200
220 XA=X
230 GOTO 175
240 PRINT X;Y
250 STOP
400 YOLD=F
410 X=XA-YA*(XB-XA)/(YB-YA)
420 GOSUB 1000
430 Y=F
440 IF ABS(XB-XA)<=E THEN 240
450 RETURN
1000 F=((X+1)^10-1)/X/(1+X)^10-
     7.889
1010 RETURN
```

The subroutine at line 1000 is User-definable and is nothing but the function F. The one given here is only an example, and if you run either program with XA=.04 or less and XB greater than .05 you will find a zero at .04561 or thereabouts, depending on the precision you choose.

The second number which appears with this answer is the value of the function F at that point.

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>END
```

PROGRAMS THAT WRITE PROGRAMS

By JIM PETERSON (continuing from V3.10)

NOTE: In these listings a £ sign should be replaced with a #

Part 2

Last month I promised you something more useful, so here it is. This routine will come in very handy for formatting screen text into neat 28-column lines, and will save the text in program lines of DATA statements. When you are ready to save, type @@@ and enter as the last line, then NEW and MERGE DSK1.LINEFILE -

```
100 !LINEWRITER to aid in fo
rmatting screen text into 28
-column format and saving it
 as DATA program lines in ME
RGE format - by Jim Peterson
110 !strings containing comm
as and quotation marks will
be ACCEPTed, and converted t
o DATA statements which RUN
correctly even though they
120 !are not enclosed in qu
otation marks!
130 CALL CLEAR :: OPEN £1:"D
SK1.LINEFILE",VARIABLE 163 :
: LN=30000
140 FOR R=1 TO 24 :: DISPLAY
 AT(R,1)SIZE(1):" " :: ACCEP
T AT(R,0)SIZE(-28):A$ :: IF
A$="@@@" THEN 180 :: B$=B$&C
HR$(200)&CHR$(LEN(A$))&A$
150 X=X+1 :: IF X/4=INT(X/4)
THEN 160 ELSE B$=B$&CHR$(179
):: GOTO 170
160 GOSUB 210 :: LN=LN+10
170 NEXT R :: X=0 :: CALL CL
EAR :: GOTO 140
180 IF B$="" THEN 200 :: IF
SEG$(B$,LEN(B$),1)=CHR$(179)
THEN B$=SEG$(B$,1,LEN(B$)-1)
190 GOSUB 210
200 PRINT £1:CHR$(255)&CHR$(
255):: CLOSE £1 :: END
210 PRINT £1:CHR$(INT(LN/256
))&CHR$(LN-256*INT(LN/256))&
CHR$(147)&B$&CHR$(0):: B$=NU
L$ :: RETURN
```

Oh - that puzzle in last month's article? Try creating those DATA statements with this LINEWRITER program!

Now, let's get down to business and learn how to do all this. First, let's write a program that will write a program to list the token codes that you need to use to write a program that will write a program -

```
100 OPEN £1:"DSK1.TOKENLIST"
,DISPLAY ,VARIABLE 163,OUTPU
T :: FOR N=129 TO 254 :: L1=
INT(N/256):: L2=N-256*L1
110 PRINT £1:CHR$(L1)&CHR$(L
2)&CHR$(131)&CHR$(N)&CHR$(0)
:: NEXT N
120 PRINT £1:CHR$(255)&CHR$(
255):: CLOSE £1 :: END
```

Key that in, RUN it, then enter NEW, then MERGE DSK1.TOKENLIST. Now LIST it and you will see a list of ASCII codes 129 through 254 and their token meanings. Delete lines 171 through 175, 185, 198, 226 through 231, and 242. Change the definition of 199 to QUOTED STRING, of 200 to UNQUOTED STRING, and 201 to LINE NUMBER, and add line 255 !END OF FILE.

You don't need all those exclamation points, so change the program to a DIS/VAR 80 file by LIST "DSK1.TOKENLIST". Then key in this little routine.

```
100 OPEN £1:"DSK1.TOKENLIST"
,INPUT :: OPEN £2:"PIO" !or
whatever
```

))))))

NOTE: When you see a £ below, use a #

```
110 PRINT £2:CHR$(27);"N";CH
R$(6)
120 LINPUT £1:A$ :: PRINT £2
:TAB(10);SEG$(A$,1,4)&SEG$(A
$,6,255):: IF EOF(1)<>1 THEN
 120 ELSE CLOSE £1 :: END
```

RUN it, and print out a list of all the token codes. Keep it handy, you'll be needing it. Notice that every Extended Basic statement has its own ASCII token code - even the ones you perhaps never heard of, such as LET and GO. Notice also that every keyboard symbol which affects program execution, such as + and =, has its own ASCII token code which is NOT the same as its keyboard ASCII code. And notice that the double colon, used as a separator in Extended Basic multi-statement lines, has its own token.

Now, let's take a look at how a MERGE format program is put together. This routine will do that for you - and you will also find it very useful in debugging the MERGE programs you are going to write.

```
100 DISPLAY AT(3,5)ERASE ALL
:"D/V 163 FILE READER": :"
    by Jim Peterson": : :" T
o edit a file saved or":"cre
ated in MERGE format."
110 DISPLAY AT(12,1):"Output
 to? (S/P)S":" (S)creen":" (
P)rinter" :: ACCEPT AT(12,17
)SIZE(-1)VALIDATE("SP"):Q$
120 IF Q$="P" THEN DISPLAY A
T(14,1):"PRINTER? PIO" :: AC
CEPT AT(14,10)SIZE(-18):P$ :
: D=2 :: OPEN £2:P$
130 DATA ELSE,":",!,IF,GO,G
OTO,GOSUB,RETURN,DEF,DIM,END
,FOR,LET,BREAK,UNBREAK,TRACE
140 DATA UNTRACE,INPUT,DATA,
RESTORE,RANDOMIZE,NEXT,READ,
STOP,DELETE,REM,ON,PRINT,CAL
L
150 DATA OPTION,OPEN,CLOSE,S
UB,DISPLAY,IMAGE,ACCEPT,ERRO
R,WARNING,SUBEXIT,SUBEND,RUN
,LINPUT
160 DATA ,,,,,,THEN,TO,STEP,"
,",";","#",),(,&,,OR,AND,XOR
,NOT,=,<,>,+,-,*,/,^,
170 DATA QUOTED STRING,UNQUO
TED STRING,LINE NUMBER,EOF,A
BS,ATN,COS,EXP,INT,LOG,SGN,S
IN
180 DATA SQR,TAN,LEN,CHR$,RN
D,SEG$,POS,VAL,STR$,ASC,PI,R
EC,MAX,MIN,RPT$,,,,,,,,NUMERI
C,DIGIT
190 DATA UALPHA,SIZE,ALL,USI
NG,BEEP,ERASE,AT,BASE,,VARIA
BLE,RELATIVE,INTERNAL,SEQUEN
TIAL,OUTPUT,UPDATE,APPEND
200 DATA FIXED,PERMANENT,TAB
,£,VALIDATE
210 DIM T$(126):: FOR J=1 TO
 126 :: READ T$(J):: NEXT J
:: E$(1)="LINE NOT CLOSED WI
TH CHR$(0)"
220 DISPLAY AT(16,1):"FILENA
ME? DSK" :: ACCEPT AT(16,14)
:F$
230 ON ERROR 240 :: OPEN £1:
"DSK"&F$,VARIABLE 163,INPUT
:: GOTO 250
240 DISPLAY AT(20,1):"I/O ER
ROR" :: ON ERROR STOP :: RET
URN 220
250 ON ERROR 260 :: LINPUT £
1:A$ :: X=ASC(SEG$(A$,1,1)):
: Y=ASC(SEG$(A$,2,1)):: IF X
=255 AND Y=255 THEN 410 ELSE
 270
260 PRINT £D:"FILE NOT CLOSE
D PROPERLY":"WITH CHR$(255),
CHR$(255) ?" :: STOP
270 PRINT £D:"LINE NUMBER":X
;"TIMES 256=";X*256:Y;"PLUS"
;Y;"=";X*256+Y
280 FOR J=3 TO LEN(A$)-1 ::
X=ASC(SEG$(A$,J,1))
290 IF X=201 THEN PRINT £D:X
;"LINE NUMBER" :: X=ASC(SEG$
(A$,J+1,1)):: Y=ASC(SEG$(A$,
J+2,1)):: J=J+2 :: PRINT £D:
X;"TIMES 256=";X*256:Y;"PLUS
";Y;"=";X*256+Y
300 IF X=199 THEN PRINT £D:X
;"QUOTED STRING" ELSE IF X=2
00 THEN PRINT £D:X;"UNQUOTED
 STRING" ELSE GOTO 360
310 J=J+1 :: X=ASC(SEG$(A$,J
,1)):: PRINT £D:X;"OF";X;"CH
ARACTERS"
320 ON ERROR 340 :: FOR L=1
TO X :: Y=ASC(SEG$(A$,J+L,1)
):: PRINT £D:Y;CHR$(Y):: IF
Y<32 OR Y>126 THEN PRINT £D:
"UNPRINTABLE CHAR - ERROR?"
```

))))))

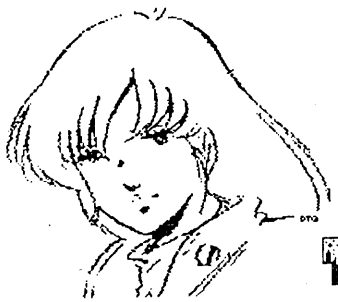
Note: The printed £s below should be #'s.

```
330 NEXT L :: J=J+X :: GOTO
370
340 PRINT £D:"ERROR! INSUFFI
CIENT BYTES IN":"STRING" ::
IF ASC(SEG$(A$,LEN(A$),1))()
0 THEN PRINT £D:E$(1)
350 ON ERROR STOP :: RETURN
250
360 IF X(129 THEN PRINT £D:X
;CHR$(X);" VARIABLE NAME" EL
SE PRINT £D:X;T$(X-128)
370 CALL KEY(0,K,S):: IF S=0
 THEN 390
380 CALL KEY(0,K2,S2):: IF S
2(1 THEN 380
390 NEXT J :: IF ASC(SEG$(A$
,J,1))=0 THEN PRINT £D:"0 EN
D OF LINE" ELSE PRINT £D:E$(
1)
400 GOTO 250
410 PRINT £D:X:X;"END OF FIL
E" :: CLOSE £1 :: STOP
```

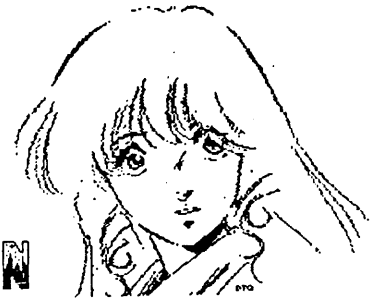
Next month - how to do it!

{Although Jim makes reference to next month and last month, in fact the previous article was published in V3.10, and the next in the series may or may not make an appearance by next month! PB}

))))))

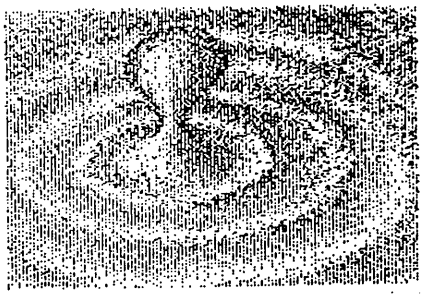


THEN



Once I was an ordinary schoolgirl...

I tasted SARSAPARILLA...

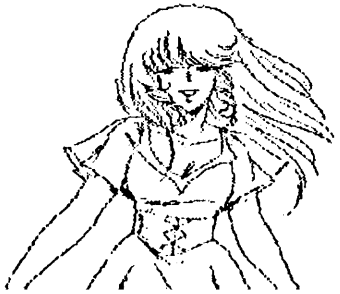


it was another world....

my life changed...

I met interesting people...

I was never the same again...



sjs

VOLUME 1: ARTICLE INDEX

A number of ITUGers have been asking for the provision of even a simple index to articles published in earlier issues of TI-LINES, so I have prepared such a list. Volumes 1 and 2 are covered, and as every issue carried an editorial and a bulletin board page, I have omitted these from the list in order to reduce the amount of space required. As you can see, just a plain list of the titles does not tell you a thing about the wide range of subjects covered by most of those articles, but it may give you some hints to follow up.

A word about pricing: because the cost of producing just one issue from the previous publications is quite high (it's not as if I was doing the usual 220 copies!) I have had to price back issues accordingly.

A single issue, regardless of size, is £2 inclusive of post and packing.

Volume 1 (14 issues) is £16 }
}V1 & V2 combined are £25}
Volume 2 (12 issues) is £14 } } V1+V2+V3 are £36
}V2 & V3 combined are £25}
Volume 3 (12 issues) is £14 }

All prices are inclusive of post and packing

Please note that any issues/volumes will have to be produced to order, but I will try to keep the delay to a minimum.

V1.1 1984 APRIL 16PP

BOOK REVIEW
Learning To Use The TI-99/4A By KEVIN TOWNSEND
THE CONTROL AND FUNCTION KEYS No.1
Examination of different uses of CTRL and FCTN
TEACH YOUR GRANDMOTHER TO SUCK EGGS DEPT
Brief notes on the Speech Separators in Extended BASIC, and CALL KEY() errors
CLOSE FILE

V1.2 1984 MAY 24PP

MODIFY YOUR MODULATOR
Connecting direct to a video recorder or a TV video socket
BEGINNER'S BASIC II No.1
Originally published in TIDINGS
THE CONTROL AND FUNCTION KEYS No.2
Tokens in TI BASIC, from the CTRL & FCTN keys

}}}

```
   MICRO TIPS
     Keys which are still "active" during OLD and SAVE to cassette
   ONES & TWOS COMPLEMENT
     Bits of binary (with bugs!)
   CLOSE FILE

V1.3      1984 JUNE              16PP

   BEGINNER'S BASIC II  No.2
   ERRATA
     Errors from V1.2 brought to light
   LETTERS
   THE CONTROL AND FUNCTION KEYS  No.3
     More on the tokens: hiding data
   BOOK REVIEW
     NEC 30 Hour BASIC By CLIVE PRIGMORE
   CLOSE FILE

V1.4      1984 JULY              24PP

   FOR THE YOUNGER TI USER
     13 year old DAVID BROWN's page
   THE CONTROL AND FUNCTION KEYS  No.4
     Further notes
   THE QUESTIONNAIRE
     Report on the summary findings
   PAUL K DUNDERDALE
     PIKADEE's proprietor on typefaces and using POS()
   BEGINNER'S BASIC II  No.3
   LETTERS
   THE SPEECH SYNTHESIZER: A DETAILED ANALYSIS  No.1
     Brief examination of CALL SPGET() and Linear Predictive Coding
   CLOSE FILE

V1.5      1984 AUGUST            32PP

   PAUL K DUNDERDALE
     'Hi-Score' routine
   BEGINNER'S BASIC II  No.4
   JAN KNAPEN
     A letter from Holland concerning BASIC tokens
   UNCOVERING UNDOCUMENTED PROGRAMS  No.1
     Follow-on from the CONTROL & FUNCTION series
   CONTACTS PAGE
   DAVID BROWN
     Young 'uns page written by a young 'un
   UTILITY
     "MiniMemory to disk" save routine from GARY HARDING
   THE SPEECH SYNTHESIZER: A DETAILED ANALYSIS  No.2
     The interpretation of bytes of raw speech data
   CLOSE FILE

V1.6      1984 SEPTEMBER         24PP

   MICRO-LINES
     DAVID BROWN by another name!
   USING YOUR JOYSTICKS  No.1
   LETTERS

))))))
```

BEGINNER'S BASIC No.5
UNCOVERING UNDOCUMENTED SUBPROGRAMS No.2
Brief details of an undocumented subprogram in TI BASIC
THE SPEECH SYNTHESIZER: A DETAILED ANALYSIS No.3
Modifying the pitch of a sound
CLOSE FILE
TABLE OF TI BASIC COMMAND TIMINGS

V1.7 1984 OCTOBER 28PP

PAUL K DUNDERDALE
Colour squares routine
BEGINNER'S BASIC No.6
DAVID BROWN
The character definitions used by Extended BASIC
ENHANCED BASIC No.1
Follow-on from Undocumented Subprograms: looking at PRK & STATS
A TASTE OF ADVENTURE
IAN MARTIN sent in a printout from INFOCOM's THE WITNESS
GENERAL DESCRIPTION OF THE ACCEPT SUBPROGRAM
From TI's own document, with permission
GENERAL DESCRIPTION OF THE DISPLAY SUBPROGRAM
From TI's own document, with permission
CLOSE FILE

V1.8 1984 NOVEMBER 28PP

DAVID BROWN
Musical musings for the Young Ones
FILE RECOVERY
Examining a disk without machine code
ENHANCED BASIC No.2
CALL P()
GENERAL DESCRIPTION OF THE PREP SUBPROGRAM
From TI's own document, with permission
MANCHESTER MEETING
A ..er.. report on the first TI-EXCHANGE National Meeting
ECHO
How to "echo" your CALL KEY() presses to the screen
USING YOUR JOYSTICKS No.2
A more detailed look at using CALL JOYST() in your own programs
CLOSE FILE

V1.9 1984 DECEMBER 24PP

UNDOCUMENTED CALLS
A series of subprograms resident on the TI Disk Controller card
MBX REVIEW
Findings on the MBX Expansion Unit (with Speech)
UTILITY
Reprint of the listing from V1.5
ENHANCED BASIC No.3
CALL H()
A CONSIDERABLE DEGREE OF MUCKING ABOUT
Musings on what happens to a program when it is OLDed from cassette
CONTACTS
GENERAL DESCRIPTION OF THE HEADER SUBPROGRAM
From TI's own document, with permission

))))))

CALL KEY REVISITED
What key unit 4 can do
CLOSE FILE

V1.10 1985 JANUARY 20PP

BOOK REVIEW
Learning TI-99/4A Home Computer Assembly Language By IRA McCOMIC
DAVID BROWN
3D cubes this time...
CONTACTS
A LETTER TO THE TI*MES
ALLEN BURT on using printer control codes in TI-Writer Editor
ENHANCED BASIC No.4
Deeper discussion of CALL H()
CLOSE FILE

V1.11 1985 FEBRUARY 24PP

CONTACTS
GARY HARDING
Discussion of Interrupts, and a Clock Listing for MiniMemory
ENHANCED BASIC No.5
Even more on CALL H()
TI-WRITER TIPS
CLOSE FILE

V1.12 1985 MARCH 20PP

CONTACTS
SYD'S SNYPPETS
From SYD MICHEL, two pages of hints and tips
DAVID BROWN
Graphics and a perspective view
LETTERS
CLOSE FILE

V1.13 1985 APRIL 60PP

CONTACTS
CENTRONICS
From MALCOLM HEDLEY and DAVE KNOTT, a hardware PIO fix by CENTRONICS
LETTERS
OTTAWA TI USERS NEWSLETTER
BRUCE CARON describes Sector Zero's Disk Bit Map
ENHANCED BASIC No.6
CALL G() for a change...
SASKATOON TI COMPUTER CLUB NEWSLETTER
The File Descriptor Block
RGB AND YOUR TI-99/4A
VIV COMLEY and his RGB interface
GENERAL DESCRIPTION OF THE GETPUT SUBPROGRAM
From TI's own document, with permission
DAVID'S COLUMN
A submission from RICHARD OWEN entitled DOODLE
BRIEF NOTES
HOWARD GREENBERG's thoughts about some products

››››››

```
   LOADS OF USE
     From the SASKATOON TI COMPUTER CLUB, a compilation of CALL LOAD()s
   PEEKS AND POKES
     As above
   ROCKY MOUNTAIN 99ERS
     Peripheral Expansion Box disassembly procedure
   GETTING THE MOST OUT OF TI-WRITER
     ALLEN BURT on the Special Character mode in the Editor
   STICC
     SASKATOON TI COMPUTER CLUB examines the storage of BASIC programs
   POWER TO THE PEOPLE
     TONY RALPH on building a simple DC power supply
   MICRO REPORT
     MILWAUKEE AREA 99/4A USERS GROUP with Forth screens
   CLOSE FILE

V1.14     1985 MAY                44PP

   BRIGHTON REPORT
     Tongue-in-cheek report on the Second TI-EXCHANGE meeting
   STICC
     Changing colours in the Editor/Assembler, By TERRY ATKINSON
   CONTACTS
   INTERFACING THE TI-99/4A  No.1
     Part 1: INPUT  by DAVE HEWITT
   DAVID BROWN
     More for the younger OTIUser
   CIN-DAY USER GROUP
     A brief resume of the disk directory sectors
   GENERAL DESCRIPTION OF THE LOAD SUBPROGRAM
     From TI's own document, with permission
   GENERAL DESCRIPTION OF THE SAVE SUBPROGRAM
     From TI's own document, with permission
   9T9 USER GROUP TORONTO
     The LOAD INTERRUPT switch, by JOHN BANNISTER
   ENHANCED BASIC  No.7
     CALLs L() and S()
   MSP 99
     Part 1 of a discussion of disks and drives
   LETTERS
   THE PHILADELPHIA AREA TI-99/4A USERS GROUP
     Cleaning your keyboard, by KEN CHANDLER
   UNDOCUMENTED FACILITIES
     Information from HOWARD GREENBERG, concerning TI-Writer
   CLOSE FILE

>>>
```

VOLUME 2: ARTICLE INDEX

V2.1 1985 JUNE 20PP

MSP 99 TI USER GROUP
Customising the TI-Writer characters, by GLENN DAVIS
MALCOLM HEDLEY
How to power more than one drive with the PEB
CONTACTS
CLOSE FILE

V2.2 1985 JULY 20PP

HOWARD GREENBERG
A review of the MYARC 128K card by the man who sells it
STEPHEN SHAW
His Public Domain offerings
MSP 99 REVIEW No.1
Navarone Disk Fixer compared with disk fixing in Forth,
By JOEL GERDEEN
MSP 99 REVIEW No.2
Adding a numeric keypad by NORMAN RIGER
CONTACTS
CLOSE FILE

V2.3 1985 AUGUST 28PP

PUBLIC DOMAIN SOFTWARE LIST
An update from STEPHEN SHAW
GETTING DOWN TO BASICS
JOHN BAIRD on the ups and downs of a TI owner
CONTACTS
RICHARD OWEN
The little debugger...
SORTING AND SEARCHING No.1
DIS/VAR WARS
The everyday story of alien folk begins here...
ENHANCED BASIC No.8
PAUL KARIS and FRANC GROOTJEN begin their discussion
DISK FILE DIRECTORY ENTRIES
BRUCE CARON examines sectors 2 to 33
FURTHER EXPLANATION
Expanding some of the details in Bruce's article
CLOSE FILE

V2.4 1985 SEPTEMBER 24PP

USABLE CASSETTE RECORDERS
A list of those that seem TI-compatible
ENHANCED BASIC No.9
PAUL KARIS and FRANC GROOTJEN with a brief summary of the commands
LETTERS
MSP 99 TI USER GROUP
DAVE WUNDERLIN's TI Forth update

)))))))

DAVID BROWN
For the younger User
DIS/VAR WARS
SAMPLE PAGES FROM 4ABC
ALAN DAVEY's sole UK BULLETIN BOARD
SORTING AND SEARCHING No.2
PRINTING WITH THE TI-99
DAVE HEWITT on accessing peripherals
CLOSE FILE

V2.5 1985 OCTOBER 28PP

CONTACTS
LETTERS
SYD MICHEL debugs MiniMemory's Line-by-line Assembler
ENHANCED BASIC No.10
PAUL KARIS and FRANC GROOTJEN on Word Processing with Enhanced BASIC
DIS/VAR WARS
AID FOR DEF USERS
Using DEF to gooD EFfect ?
MSP99 TI USER GROUP
MIKE KABALA comes Forth with some music
RICHARD OWEN
Making flippies
SORTING AND SEARCHING No.3
Flag making and tasting Bubbles
BULLETIN BOARD LIST
BBSs in the Denver area
OTTAWA TI USER GROUP
LLOYD GALENZOSKI on using the 24K RAM space as a file
CLOSE FILE

V2.6 1985 NOVEMBER 36PP

MSP 99 USER GROUP
BOB HUBEL's modification to quieten the PEB
DIS/VAR WARS
A fractured file filled with fractured thoughts
RICHARD OWEN
Richard presents a precis of GARY HARDING's original Assembly Language articles
OTTAWA TI USER GROUP
BRUCE CARON on adding 8K to Editor/Assembler, and accessing disks from Assembly Language programs
ENHANCED BASIC No.11
PAUL KARIS and FRANC GROOTJEN present an application in Enhanced BASIC
LETTERS
DAISEY WHEEL writes to DOT MATRIX
SOFTWARE FROM STEPHEN SHAW
Stephen's holdings of Public Domain software
SORTING AND SEARCHING NO.4
Presenting a sample Bubble-sorting routine
INPUT VALIDATION or MUG TRAPPING
One for newcomers, in console BASIC
CONTACTS
FORTH BUGS
From the MILWAUKEE AREA 99 USER GROUP

))))))

SHOW REPORT
  Ol' Baldie on Birmingham's TI-EXCHANGE meeting
CLOSE FILE

V2.7 1985 DECEMBER 28pp

COMBINING EDITOR/ASSEMBLER, DISK MANAGER, & TI-WRITER
  RICHARD BAILEY on making a Supermodule
SILENCING THE BOX
  JOHN ROE with a cautionary note about modifying the PEB fan
PSEUDOCODE
  A brief look at this useful programming technique
DISK DRIVES
  A discussion by JIM NESS via LLOYD GALENZOSKI
MYARC DISK CONTROLLER CARD
  ALLEN BURT conducts an in-depth User's review
STEPHEN SHAW's SOFTWARE COLLECTION
  An update on Stephen's holdings
A LOOK AT GPLLNK
  R. A. GREEN discusses and presents his routine to link to GPL
LETTERS
A LOOK AT USING GPLLNK
  R. A. GREEN presents applications for his GPLLNK routine
CLOSE FILE

V2.8 1986 JANUARY 40pp

JAMES STRINGFELLOW
' No less than 8 called-from-BASIC machine code routines to try out
DIS/VAR WARS
  Peppered throughout this issue
PLOTTING A SINE WAVE
  A lengthy discussion on turning Sines into Waves
CHANGING XB EDITING COLOURS
  A continual interruption which could make you see RED. Or CYAN.
  Or MAGENTA...
LETTERS
MAXIMEM
  A brief review of a GRAMKRACKER competitor
CLOSE FILE

V2.9 1986 FEBRUARY 24pp

STARTING FORTH No.1
A LOOK AT ALPHALOCK
  A short machine code routine to scan the alphalock key
SPECIAL FEATURE
  "MATCHBOX" 32K RAM EXPANSION
I BUILT A MATCHBOX RAM EXPANSION
  GRAHAM WOLSTENHOLME on his experiences
HINTS ON BUILDING A MATCHBOX EXPANSION
  Tips passed on by GRAHAM WOLSTENHOLME
CLOSE FILE

V2.10 1986 MARCH 36pp

CONTACTS
JAMES STRINGFELLOW
  A "graphic" example of the application of TI-Writer
))))))

SORTING AND SEARCHING No.5
Tag sorting
MODULES/CASSETTES/DISK/BOOKS
A list of those available through OTIU
KEYBOARD SCHEMATIC
CONFIGURING FORTH I No.1
CONFIGURING FORTH II No.2
A LOOK AT PROGRAMS
An examination of different structures of stored programs
DIS/VAR WARS
SMILING ENIGMA
Caption competition
CLOSE FILE

V2.11 1986 APRIL 40PP

FLOWER POWER IN FORTH
Plotting curves using Forth
SERVICES
Offered by WEST MIDLANDS TI USERS
PRINT AT IN TI BASIC
Imitating PRINT AT using console BASIC
LETTERS
Accessing PRESTEL
DRIVE PROBLEMS
Caused by a faulty PEB interface card
TI-WRITER PLUS
PAUL MEADOWS on printing beyond 80 columns
CENTRONICS FROM RS232
Fool your RS232 into becoming CENTRONICS
WATCHING A PROGRAM LOAD
OLDing monitor in TI BASIC
LEARNING FORTH No.3
BEGINNING FORTH No.4
SORTING AND SEARCHING No.6
Ripples, shakes...
CAPTION COMPETITION
Reprint of the Smiling Enigma
CLOSE FILE

V2.12 1986 MAY 28PP

LEEDS SHOW REPORT
RAMDISK DATA
Advertising the New Horizons RAMDisk
IN MEMORIAM THOMAS WEITHOFER
MONITOR SPECIFICATIONS
The monitor supplied through RICHARD SIERAKOWSKI
LETTERS
NOTES FROM STEPHEN SHAW
C99 REVIEW
CLOSE FILE

A complete subject index is being prepared by PETER KILLICK and will cover volumes 1, 2, and 3. The index here is simply a very brief guide to what has been published in volumes 1 and 2.

))))))

N O T I C E B O A R D

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

Quite a few things up for grabs this month, and an exchange is offered by ITUGer KEN WILFORD, who would like a TI Expansion System and is putting up a Commodore 64, two cassette decks, a backup board, joysticks and a considerable quantity of other material: eleven books, six C64 cartridges (including Lazarian, Ms Pacman, and Wizard of Wor), four book-plus-tape packs, no less than fifteen utilities (including Simons BASIC which, if memory serves correctly, demonstrated how Commodore could be bested by a fifteen year old lad!), nineteen Adventures (including the Hobbit, Lord of the Rings, Valhalla, and many others), and over 100 other good games with an equal number of "extras". Someone looking to opt out of the TI side of things might like to consider the C64 - I know a number of ITUGers have them as alternatives to the 4A.

Contact Ken on Chester 373392 if you are seriously interested.

DEREK ALLEN (see also LETTERS) has asked if any ITUGers in the CORNWALL area would like to contact him with a view to getting together and maybe seeking out other 4A owners who have not yet enjoyed the benefits of mutual support. Write to him at 17 Bank Street, St Columb Major, Cornwall, TR9 6AT if you are interested.

ANDY MACMAHON in the Republic of Ireland is eager to obtain a TI Disk Controller card. He has deposited cash with a UK-resident ITUGer to expedite a quick sale at the right price. Either call Andy using the code sequence 010 353 21 028 38376 (if I've worked the codes out right!) or give your details to me and I will pass them on.

MUKUND RAJPARA has a few items for sale, prices for which I do not have at the time of writing. Contact Mukund, if you have an offer to make, at 10 April Croft, Moseley, Birmingham, B13 9HP:

WHSmith Data Recorder. Navarone Cartridge Expander. Speech Synthesizer. TI Peripheral Expansion System with all cards. Cumana Single-sided disk drive. Olivetti JP101 Ink Jet Printer with all attachments.

)))))))

2023 note: This list is of items for sale from the group to group members.

TITLE OF ITEM	FORM	PRICE	STOCK LEFT, AND COMMENTS
ABM CONTROL/CAVERN HUNT	CASS	2.00	7
ABM CONTROL/FROGLET	CASS	2.00	2
ADDITION & SUBTRACTION 1	CART	5.00	1
ADDITION & SUBTRACTION 2	CART	5.00	1
ALIEN ADDITION	CART	5.00	8
ALLIGATOR MIX	CART	5.00	6
ALPINER	CART	9.00	1
ASCOT STAKES/TORPEDO ALLEY	CASS	2.00	5
BASIC GAMES PACK 1	CASS	2.00	1
BATTLESTAR ATTACK	CASS	2.00	1
BEGINNING GRAMMAR	CART	5.00	1
BLACK BOX	CASS	1.50	1
BLASTEROIDS	CASS	2.00	3
BOMBS AWAY	CASS	2.00	8
BOUNCER	CASS	2.50	3
BURGERTIME	CART	9.00	2
CHARACTER GENERATOR	CASS	2.50	7
CHECKBOOK MANAGER	DISK	6.50	1
CONNECT FOUR	CART	7.00	1
DADDIE'S HOT ROD	CASS	2.00	1
DEFENDER	CART	9.00	1
DEMOLITION DIVISION	CART	5.00	5
DEMONSTRATION	CART	3.50	2
DIYAD	CASS	2.00	4
EARLY READING	CART	5.00	3
ESPIAL	CART	9.00	4
FACEMAKER	CART	9.00	2
FROGLET/BATTLESHIPS	CASS	2.00	2
FUN-PAC	CASS	2.00	1
GRAPHICS CREATOR	CASS	2.00	1
HOP-IT	CASS	2.00	3
HOPPER	CART	9.00	1
HOUSEHOLD BUDGET MANAGEMENT	CART	6.50	2
INDOOR SOCCER	CART	8.50	1
METEOR MULTIPLICATION	CART	5.00	7
MINUS MISSION	CART	5.00	5
MISSILE ALERT	CASS	2.00	1
NUMERATION 2	CART	4.50	1
OPERATION MOON	CASS	2.00	5
OTHELLO	CASS	2.00	9
PARSEC	CART	9.00	6
PEARL DIVER	CASS	2.00	9
PERSONAL FINANCIAL AIDS	CASS	4.50	1
PERSONAL REPORT GENERATOR	CART	6.50	2
PROGRAMMING AIDS 1	CASS	6.50	2
QUASIMODO HELP	CASS	5.50	1
RETURN TO PIRATE ISLE	CART	9.00	1
ROBOPODS	CASS	2.00	3
RUNNER ON TREITON	CASS	2.00	1
SPEECH EDITOR	CART	9.00	1
STATISTICS	CART	11.00	1
STORY MACHINE	CART	9.00	2
SUPERFLY	CART	11.00	1

(continued).....

TITLE OF ITEM	FORM	PRICE	STOCK LEFT, AND COMMENTS
TAX/INVESTMENT RECORD KEEPING	CART	9.00	1
TEACH YOURSELF BASIC	CASS	2.50	1
TEACH YOURSELF EXTENDED BASIC	CASS	2.50	1
TOAD GRAPHICS	CASS	2.50	2
TROLL KING	CASS	2.00	1
ZERO ZAP	CART	6.50	4
TI SOUND & GRAPHICS	BOOK	8.00	2
33 PROGRAMS FOR THE TI	BOOK	8.00	2
CREATING ARCADE GAMES	BOOK	8.00	1
EXT'D BASIC HOME APPLICATIONS	BOOK	8.00	1
TI COLLECTION VOL. 1	BOOK	8.00	1
SPECIAL OFFERS			
STARTER PACK 1	BK/CS	3.00	8 }
STARTER PACK 2	BK/CS	3.00	2 } ALL LABELLED BK/CS ARE
GAMES WRITER PACK 1	BK/CS	3.00	1 } BOOK AND CASSETTE PACKS
GAMES WRITER PACK 2	BK/CS	3.00	1 }
ADVENTURE PROGRAMS FOR USE WITH THE MODULE			
ADVENTURELAND	CASS	3.00	1
GHOST TOWN	CASS	3.00	1
MISSION IMPOSSIBLE	CASS	3.00	1
MYSTERY FUN HOUSE	CASS	3.00	1
THE GOLDEN VOYAGE	CASS	3.00	1
VOODOO CASTLE	CASS	3.00	1

Please note that you MUST have current ITUG membership in order to take advantage of this list

Orders totalling £10 or more are post-free

Orders less than £10 will have post, packing, and insurance added

Please note that we do try to keep this list up to date, but sometimes the items you request may not be available; we will either offer you a replacement, or credit you against future purchase, or offer you a refund of the outstanding amount.

THE TIHOME SOFTWARE COLLECTION

Pricing for TSC disks has been reorganised, although the full Collection is still offered at £35 inclusive.

Check the TSC Catalogue (available free to ITUG subscribers on request) to find out what the TSC entries stand for. Make sure that you indicate clearly what your choice is, specifying the name of the disk/s you want.

GAMES:	TSC ENTRIES:
TSC_DSK_A	GA0001 - GA0011
TSC_DSK_B	GA0012 - GA0022
TSC_DSK_C	GA0023 - GA0032
TSC_DSK_D	GA0033 - GA0043
TSC_DSK_E	GA0044 - GA0055
TSC_DSK_F	GA0056 - GA0066
EDUCATIONAL:	
TSC_DSK_G	ED0001 - ED0009
TSC_DSK_H	ED0010 - ED0017
DEMONSTRATION:	
TSC_DSK_I	DE0001 - DE0006
MUSIC:	
TSC_DSK_J	MU0001 - MU0009
TSC_DSK_K	MU0010 - MU0017
TSC_DSK_L	MU0018 - MU0026
UTILITIES:	
TSC_DSK_M	UT0001 - UT0011
TSC_DSK_N	UT0012 - UT0022
TSC_DSK_O	UT0023 - UT0033
TSC_DSK_P	UT0034 - UT0044

PRICING

The number of programs on a disk can vary between about 7 and 11, dependent upon the sizes of the programs in terms of sectors used.

The exception is the DEMONSTRATION disk, which alone is offered at £2.95 inclusive of post and packing

The rest are priced as follows:

£ 3.95	for ANY 1 DISK
£ 6.90	for ANY 2 DISKS (SAVE £ 1)
£ 9.85	for ANY 3 DISKS (SAVE £ 2)
£12.80	for ANY 4 DISKS (SAVE £ 3)
£14.75	for ANY 5 DISKS (SAVE £ 5)
£16.70	for ANY 6 DISKS (SAVE £ 7)
£18.65	for ANY 7 DISKS (SAVE £ 9)
£20.60	for ANY 8 DISKS (SAVE £11)
£22.55	for ANY 9 DISKS (SAVE £13)
£24.50	for ANY 10 DISKS (SAVE £15)
£26.45	for ANY 11 DISKS (SAVE £17)
£28.40	for ANY 12 DISKS (SAVE £19)
£30.35	for ANY 13 DISKS (SAVE £21)
£32.30	for ANY 14 DISKS (SAVE £23)
£34.25	for ANY 15 DISKS (SAVE £25)
£35.00	ENTIRE (SAVE £27.20)

All programs are recorded on Single-sided disks.

You may elect to be supplied with software on Double-sided disks, when you should deduct £1 from the prices shown above.

If you prefer, you may send in your own disks, in which case deduct 50p for each disk from the prices shown above.

For example, 8 disks supplied by you would work out at £20.60 - (8 x 50p = £4) = £16.60 nett.

All the above prices are inclusive of post and packing.

The pricing for programs recorded on cassette remains £1 per program, with an overall charge of 65p for post and packing.

PLEASE MAKE ALL CHEQUES PAYABLE TO "PETER BROOKS"