INTERNATIONAL TI-LINES

MEVAGISSEY HARBOUR

RICHARD THYNING

Magazine scanned 2022 by Stephen Shaw

Formerly OXON TI USERS

A PUBLICATION OF THE INTERNATIONAL TI USER GROUP

Index to articles

Graphic images submitted by RICHARD TWYNING and peppered throughout the
issue. The Fly may lurk on some pages, but there are no flies on the
authors!

(Sorry, couldn't resist that one, I'll come quietly, promise...)

Numbers update: for those who like statistics, our Merrie Thronge are
currently around 250 in size and growing slowly (very slowly) but at
least we ARE still growing!

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

A Happy New Year to one and all, and I hope that your Christmasses were
as quiet and peaceful as mine (eventually) was. I originally intended
to be away from the 11th of December through to the 9th January, but as
usual the best laid plans of mice, men, and Pete Brooks etc., etc., came
to naught, and I had to be content with a few days around the 11th., and
a few days around Christmas and the New Year.

Several unforeseen problems arose, both at work and at home, resulting
in unavoidable delays in sending out both the magazine and other items,
but I hope that very few people were inconvenienced to any great extent.
As a result of changes at work, my free time was reduced considerably,
and I frequently found myself with many messages on the answering
machine and no time to respond to them (unless any of you are able to
accept calls from me at 1 am ?).

January and February are going to be tight as far as time schedules go,
and as I am only now writing this editorial on the 18th. of January, it
is a foregone conclusion that I will have to issue January's TI-LINES
with February's if I am to try and keep to the standard timetable.

It is not for lack of material that I am pushed to keep up with my own
deadlines, since I have masses of raw articles upon which to draw, but
it all takes time, and a five page article can take a week to check
fully (and even then I seem to miss some obvious clangers). What I
really need is a small but regular and reliable supply of material which
is all but proofread, so that I have as little editorial work to do as
is possible. In an ideal world...

It may all come out in the wash, but I fear that the laundry is going to
be a month late!

Once again the postal service has been playing pranks. So far, quite a
large number of TI-LINES have gone missing, as have almost a dozen ROMOX
cartridges. The intended recipients feel that the season is to blame,
in that the Post Office have a larger workload than usual. However, I
would expect the packages to be delayed by such an increase in volume,
not lost altogether. Increasingly, I hear of letters and parcels which
have eventually found their way to their destinations via some pretty
amazing routes - sometimes involving a trip from one end of the country
to the other - so perhaps I am being paranoid in thinking that the
gremlins are only plaguing me.

Anyway, if you have NOT received a particular issue of TI-LINES, then
PLEASE let me know and I will send you another.

While I think of it, if you should change your address and you write to
let me know, PLEASE include your old address so that I can make sure

>>MORE)

I make the changes to the correct record in my file. I recently had a subscriber tell me he'd moved, but he didn't tell me from where, and he had a counterpart with the same name in a similar area. In the end, I compared handwriting from letters in my files to determine exactly who it was. Next time I might not be so fortunate.

I appear to have lost contact with Dutch subscriber PAUL KARIS — at least, his last few magazines have been returned without being delivered (and he is usually very good about updating his changes of address).

If anyone knows where he is at present, would you be kind enough to ask him to contact me ? There have been cases in the past when correctly-addressed post has been returned because of incompetence on the part of the Post Office, but in this case it is the Netherlands Post Office which is returning TI-LINES.

On the inside cover of the last issue, I mentioned that DM1000 V3.5 had wreaked havoc on some files, and caused a major headache at the last minute before publication.

Lest anyone connected with DM1000 should feel that this is a slur upon their character, I should point out that I did not intend to attack DM1000 at all.

DM1000, like most high-speed disk handling programs (Myarc excepted) achieves its speed in writing to disk by reducing the amount of time spent in verification. Providing Users recognise this (and I usually do, but I was in a hurry), DM1000 is a marvellous piece of Public Domain software. It is far more User-friendly than TI's Disk Managers, and so much easier to work with.

However, the reduction in validation can sometimes cause problems. The most common involves an incomplete validation of a faulty sector, which can fool DM1000 into thinking that the sector is OK when it most certainly is not.

This can arise because a byte-by-byte check of a written sector is not performed; DM1000 simply reads the written sector and checks the disk controller's inbuilt error flag to see if there were any problems in reading the sector.

It is possible for a sector to be read in without causing errors, while failing to have been recorded correctly. Programs like TI's Disk Managers then go on to compare the recently written sector byte by byte with the sector intended to be written.

A daft analogy might be to contrast the enquiry "Can you hear me ?" with the enquiry "Can you understand what I say ?". You may be able to hear someone, but not necessarily understand what they are saying. This is the essential difference between programs like DM1000 and Disk Manager.

In the case in question, DM1000 confused a number of files, mixing them up (possibly because they were fractured at the time), and losing large sections of a couple of files, which then had to be retyped.

>>>MORE)

I don't have a record of the exact circumstances under which the errors occurred, otherwise I would notify the distributors of DM1000 so that they could search for the source of the errors. I usually make backups of all my disks using Disk Manager 2, precisely because it may be slow, but I can rely on it; in this case, I backed up using DM1000, and then wrote the backed up disk over the original (done to remove fractured files and to give me two recently-recorded disks whose magnetic domains should be fresh).

I usually perform a file copy with Disk Manager 2, but because time was pressing, I finished editing the last few files, and then, because it was convenient, used DM1000 to file-copy both disks.

I still use DM1000 because it is more efficient than Disk Manager 2 and more friendly, so don't go away with the impression that I am giving DM1000 a bad reference; as long as you work within the limitations of reliability, you should experience no major difficulty.

STOP PRESS: PLEASE DELETE "SHAMUS" FROM THE DI LIST GIVEN IN V4.7 (PAGE 10). ENTRY 0014 WILL BE FILLED AT A LATER DATE.

With reference to my comments last issue regarding the FUNNELWEB authors and their unfair treatment, I have further been told that their own User group refused to allow them to publish a rebuttal of the American criticisms. As a result, I gather that Tony resigned from the group committee and will not participate again until the group stops bending to pressure from the Americans (not, I hasten to add, ALL Americans. From what I have read and heard, few Americans agree with the personal attacks vented on the Australians).

As a recipient myself of undue and unfair pressure from a commercial enterprise which I had the temerity to openly criticise (see A MATTER OF INTRIGUE in a much earlier Editorial) - and who deserved the criticism, and I am more than willing to justify my criticisms at any time and to anybody - I can sympathise with the Australians' situation.

If Tony McGovern would like to submit his original article, and his subsequent rebuttal, for publication in TI-LINES, I would be more than happy to accept it. I have read part of the original article, and I see no justification for the villification which was heaped upon his head and upon that of his son, Will, and if their own group won't permit them the free forum from which to speak, then it should be shamed into so doing. The best way to achieve that is to have other groups publish the rebuttal.

It isn't often that I permit external issues to penetrate the usual content of TI-LINES, but this is one case which deserves inclusion.

One of the problems I face (and so, it seems, do editors the world over) is that I never know EXACTLY what certain readers would like to see.

The "certain readers" are those whose knowledge of computing is very

))MORE)

scant but who would like to read articles in TI-LINES which increase their understanding of specific items. For "beginners", such articles could start from first principles (describing the concepts involved, and the hardware used) and build up a basic foundation of knowledge in this fascinating field, upon which the individual can continue to build for themselves.

For the slightly more experienced, articles which either explain certain knotty aspects which have eluded comprehension, or which lay the basic groundwork for further learning, will probably be better received.

However, apart from one or two isolated cases, the TI-LINES readership seems content to read what it gets, rather than asking for specific details or specific subjects to be covered.

I am sure that this is not the case, but I have to rely on telepathy to try and decide what readers want to see. I have asked on a number of occasions (and I recently re-read some of my early "For Pete's Sake" articles for TIDINGS back in 1981 to see the same pleas!) for readers to let me know specific subjects that they want covering, without response.

I'm quite happy to fill TI-LINES with things that interest me, but I would much rather put in things which interest YOU (within limits - I have tried to persuade some of our more erudite subscribers to pass some of their immense knowledge and experience on to the rest of us, without success, I'm ashamed to say), so how about making a short list of half a dozen things which you don't understand, and sending them to me (or putting them on my answering machine!).

Remember, it's FINE DETAIL I need, not general subjects like LANGUAGES, GRAPHICS, FILE HANDLING, and so on.

Have you understood ALL of the commands in BASIC ? Do you understand what the difference between ROM, RAM, and GROM is ? Are WIRED REMOTE CONTROLLERS and JOYSTICKS one and the same thing ? What does the RS in RS232 stand for ? When is PARALLEL not CENTRONICS ? How can you have a computer language whose programs don't need line numbers ? Just what does the CTRL key do ?

I am sure that a large number of readers COULD prepare lists as long as my arm (I am sure, because occasionally one or two pluck up courage to ask me all sorts of questions at Workshops, and I think to myself: "Why haven't you asked me before ?" - not that I object to being asked, but it makes me wonder how many others are keeping their questions and their confusion to themselves), so once again I am pleading with readers to let me know specifically what they don't understand, so that I can try to shed some light.

It does work, occasionally. A little while ago, I wrote about shuffling and it generated a burst of activity which has since subsided. In the process, I learned something (and I am sure that others did, too). What I also learned was that some of the shuffling processes had been sorted out by people several years ago, but they had assumed that everybody else already knew all there was to know. Not so (especially in my case) as the subsequent articles and letters show.

So, take pity on an ageing relic of an editor, and write to him. You can be anonymous if you wish - as long as you contribute a question!

>>END

By JIM PETERSON

Part 3

2022 note: When you see £ below,
instead use # as in PRINT #1:
The errant use of £ was by Peter,
not by Jim.

Let's start learning how to actually write a program that writes a program.

A MERGEd program is a D/V 163 file, so -
OPEN £1:"DSK1.(filename),VARIABLE 163,OUTPUT

Every program line begins with a line number, of course. In MERGE format the line number, whether 1 or 32767, is squished into two characters. We don't need to get into how this is done, but you can accomplish it with CHR$(INT(LN/256))&CHR$(LN-256*INT(LN/256)), where LN has ben predefined as the line number.

To print a statement or command, anything that is represented by a token in the token list, just print the CHR$ of its token ASCII. For instance, the token for DATA is 147, so you would print CHR$(147).

To print a variable name, either numeric or string, just enclose it in quotes, "A" or A$".

To print a value, or a string which is not in quotation marks (such as in a DATA statement), or the word which follows a CALL, you must print CHR$(200) followed by a token giving the number of characters to follow, such as CHR$(5) for a 5-letter word such as CLEAR, then the value in quotes. For instance, the token for CALL is 157, so CALL CLEAR is CHR$(157)&CHR$(200)&CHR$(5)&"CLEAR".

Similarly, tokens for parentheses are 183 and 182, so the variable name A(1) is "A"&CHR$(183)&CHR$(200)&CHR$(1)&"1"&CHR$(182).

A quoted string is handled in the same way except that it is preceded by token 199, so PRINT "HELLO" is CHR$(156)&CHR$(199)&CHR$(5)&"HELLO". Don't worry about the quotation marks, the computer will handle that.

If you need to refer to a line number, as in GOTO 500, use token 201 followed by the line number formula, thus CHR$(134)&CHR$(201)&CHR$(INT(500/256))&CHR$(500-256*INT(500/256)).

Don't print more than 163 characters in a record. You can print multiple-statement XBasic lines, but be sure to use the double-colon token 130 as the separator, not two of the 181 colon tokens.

Each program line must end with CHR$(0) as the end-of-line indicator, and the last record you print must be CHR$(255)&CHR$(255) as the end-of-file indicator.

If you get an I/O ERROR 25 when you try to merge your program, it means that you left off the final double-255. If the program merges, but crashes when you run it, you will probably be able to spot an obvious error in the line when you LIST it. If the line looks OK but gives you a DATA ERROR or SYNTAX ERROR, you left off a CHR$(0) or gave the wrong count of characters after token 199 or 200. The program published in Part 2 will help you to track down these bugs.

Now let's write a program. What

>>>MORE)

is the longest possible one-liner program?

Well, RANDOMIZE is the longest statement that can stand alone. It is represented by the single token 149, and to repeat it must be followed by the double-colon token 130. Since any line number will take two bytes, let's use a 5-digit line number. And don't forget that final CHR$(0). That still leaves us 160 of the 163 bytes, so we can repeat tokens 149 and 130 for 79 times, followed by a final 149.

```
100 OPEN £1:"DSK1.LONG",VARI
ABLE 163,OUTPUT
110 FOR J=1 TO 79 :: M$=M$&C
HR$(149)&CHR$(130):: NEXT J
:: M$=CHR$(254)&CHR$(254)&M$
&CHR$(149)&CHR$(0):: PRINT £
1:M$ :: PRINT £1:CHR$(255)&C
HR$(255)
```

120 CLOSE £1

RUN, NEW, MERGE DSK1.LONG and LIST - over 34 lines long! But that one-liner doesn't do anything, so try this one -

```
100 OPEN £1:"DSK1.LONG",VARI
ABLE 163,OUTPUT
110 FOR J=1 TO 52 :: M$=M$&C
HR$(162)&"X"&CHR$(130):: NEX
T J :: M$=CHR$(254)&CHR$(254
)&M$&CHR$(162)&"X"&CHR$(0)::
 PRINT £1:M$
120 PRINT £1:CHR$(255)&CHR$(
255):: CLOSE £1
```

Again RUN, enter NEW, then MERGE DSK1.LONG, then RUN. You'll get a message BREAKPOINT IN 32510 (don't ask me why!) but just enter RUN again.

Next time - using DEF to make it all easier.

>>END

PROGRAMS THAT WRITE PROGRAMS

By JIM PETERSON

Part 4

Well, if you have tried your hand at any MERGE format program writing, you have already discovered that it is slow work, and you need to cram more onto a line than will fit. When a little CALL HCHAR(24,12,32,5) turned into CHR$(157)&CHR$(200)& CHR$(5)&"HCHAR"&CHR$(183)&CHR$(2 00)&CHR$(2)&"24"&CHR$(179)&CHR$(200)&CHR$(2)&"12"&CHR$(179)&CHR$ (200)&CHR$(2)&"32"&CHR$(179)&CHR $(200)&CHR$(1)&"5"&CHR$(182) you gave up? There is an easier

way! Using DEF can make the job so simple that you might decide to do all your programming in MERGE format - well no, it's not quite that easy.

The DEF does slow up program execution time considerably, especially when DEFs call each other, but we can tolerate that here.

For instance, that complicated mess of parentheses to squish a line number can be written just

>>MORE>

once as DEF LINES$(X)=CHR$(INT(X/256))&CHR$(X-256*INT(X-256)) and then, whenever you need a line number, just write LINE$(100) or whatever.

The flag token and counting of characters and all for an unquoted string can be DEF'd as U$(X$)=CHR$(200)&CHR$(LEN(X$))&X$ Then, to write "HELLO" just write U$("HELLO") and let the computer do the work. For a numeric value in the unquoted string, use UN$(X)=CHR$(200)&CHR$(LEN(STR$(X)))&STR$(X), and then 999 becomes UN$(999).

CALL HCHAR can be DEF HCHAR$=CHR$(157) for CALL and, since one DEF can call another, U$("HCHAR") and, since it is always followed by an opening parentheses,

CHR$(183) - but wait, let's define that open parentheses as OP$=CHR$(183).
Now DEF HCHAR$=CHR$(157)&U$("HCHAR")&OP$, and you can use HCHAR$ for CALL HCHAR(.

Let's also DEF the comma with DEF C$=CHR$(179) and the closing parentheses with DEF CP$=CHR$(182). Now that long HCHAR that had you discouraged can be abbreviated to CHAR$&UN$(24)&C$&UN$(12)&C$&UN$(32)&C$&UN$(5)&CP$.

I have written a program of 162 of these DEFs, and another program to print out a handy look-up chart of them. It would take 4 pages to print them, so if you want them just ask me for a copy.

>>>END

PROGRAMS THAT WRITE PROGRAMS

By JIM PETERSON

Part 5

In addition to writing programs in MERGE format, the same techniques can be used to analyze or modify programs which have been SAVEd in MERGE format. The D/V 163 file editor in Part 2 of this series was an example.

Here is a simple program to remove REM statements -

```
100 DISPLAY AT(3,5)ERASE ALL
:"REM REMOVER": : :"Program
must be SAVEd in":"MERGE for
mat by":"SAVE DSK(filename),
MERGE"
110 DISPLAY AT(12,1):"FILENA
ME? DSK" :: ACCEPT AT(12,14)
:F$ :: DISPLAY AT(14,1):"NEW
 FILENAME? DSK" :: ACCEPT AT
```

2022 note: Use OPEN #1 - don't use OPEN £1 !!!
In all cases replace £ with #

```
(14,18):NF$
120 OPEN £1:"DSK"&F$,VARIABL
E 163, INPUT :: OPEN £2:"DSK"
&NF$,VARIABLE 163,OUTPUT
130 LINPUT £1:M$ :: A=POS(M$
,CHR$(131),1):: B=POS(M$,CHR
$(154),1):: A=MAX(A,B):: IF
A=3 THEN 150 :: IF A=0 THEN
PRINT £2:M$ :: GOTO 150
140 PRINT £2:SEG$(M$,1,A-1)&
CHR$(0)
150 IF EOF(1)<>1 THEN 130 ::
 CLOSE £1 :: PRINT £2:CHR$(2
55)&CHR$(255):: CLOSE £2
```

The REM statement will begin with either a !, which is CHR$(131), or REM which is

>>MORE)

CHR$(154). So, line 130 reads in the lines one at a time. A finds the position in the line of ! and B finds the position of REM; one or the other, or both, will not be present and will equal 0. Then MAX finds the larger of A and B, which will be whichever one is present, or 0 if neither.

If ! or REM is in the 3rd position, immediately after the 2-byte line number, we want to delete the line entirely, so we do not reprint it. If A=0 then neither ! nor REM is present, so we reprint the entire line in the new file.

Otherwise, the REM statement is obviously a tail remark, so we reprint to the new file the segment of it starting with the first character and consisting of the number of characters one less than the position of the ! or REM. And, since we have lopped off the end of the line, we do not forget to replace the end-of-line marker CHR$(0).

If we have not reached the end of the file, we go back for the next line. Otherwise, we close the old file, but we remember to add the end-of-file marker to the new file before we close that too.

>>>END

L E T T E R S

A couple of people (STEPHEN SHAW and JOHN BINGHAM) wrote to advise me
that I did NOT create the word MONG (as appeared in V4.5 TI-LINES).

Stephen tells me that his Collins dictionary says the word is used in
sentences such as: "Hey cobber, get that mong off my grass" or "Mongs
may be quite friendly but sometimes bite.".

On the other hand, John, currently braving the Norwegian winter, tells
me that in Norway the word Mong has achieved an infamous definition as a
measure of currency. On a new refinery, a significant escalation of
costs has been experienced. As the refinery is at Mongstad, this has
given rise to "How much will it cost ? - About a Mong", where a Mong is
equivalent to about 3.6 billion Kroner.

Funny, I could have sworn that Kroner was an Australian margerine...

PHIL TROTTER wrote with some observations about the Music Maker module:

"On page 24 of the November issue of TI-LINES, the Music Maker module is
mentioned. My module gives the option of saving to disk or . cassette,
and of course reloading from both, so I set up a music file to see what
I could find out.

First of all, the numbers on the bottom of my module are:

   4378168 ATA 2982 USA

The file saved to disk is in DIS/FIX 128 format and always has 116
records in it. It is not possible to load the file into the console as
you will run out of memory after 87 files (95 after CALL FILES(1)).

Perhaps someone would be able to find an answer but the easy solution
must ne to buy a used module with disk capabilities.".

Phil then went on to discuss saving part of a program:

"Also on the same page, saving part of a program is mentioned. What I
do is to use an EXTRACTOR program to extract the lines I want to save,
and a little later I have a file on disk in MERGE format. The EXTRACTOR
I use is available from STEPHEN SHAW. I suppose that this is equivalent
to what you said in your reply, but the programs are already written and
ready to use.".

JOHN BINGHAM also commented on the facility of saving part of a listing.

"Saving only part of an Extended BASIC program (or TI BASIC program
running under Extended BASIC) is easy, assuming a disk system and one

>>MORE>

of the following:

1. Programming Aids III
2. Jim Peterson's utility
3. Super 99 magazine utility
4. GramKracker Extended BASIC

I expect that the new Super Extended BASIC module will also work, and this would not require a disk system. In each case, the unrequired lines are deleted and the remainder saved. I have used them all. (1) and (4) will also allow copying, moving, and partial resequencing.".

Finally, STEPHEN SHAW also wrote with some comments on saving part of a BASIC listing.

"SAVING PART OF A BASIC OR EXTENDED BASIC PROGRAM:

1. LIST to disk. e.g., LIST "DSK1.PROGRAM"
2. Use TI-Writer to edit out the lines - it may often be faster to extract segments using PrintFile than to delete. E.g., PF with 340 350 DSK1.EXTRACT
3. Go back to Extended BASIC and load TEXT LOAD by CURTIS PROVANCE. Load the text into memory quickly, then save it back out as a program.

Text load is available from me. It is a machine code program which loads as an Extended BASIC program. A simple CALL LINK() will make it read a DV80 file and "crunch" it just as though it had been typed in at the console. It is remarkably fast. It can be used to get programs into memory from text files, and/or to do commands. For instance, a text file might be:

```
CALL FILES(2)
NEW
100 PRINT "THIS IS A PROGRAM"
110 GOTO 110
CALL CLEAR
PRINT "PROGRAM ABOUT TO RUN..."
RUN
```

2022 Update: Today I would use Extractor by George Steffen   -s shaw

and, as it is not too fussy about spaces, you can now write your BASIC programs in a "structured" way:

```
100 FOR I=1 TO 100
110     FOR T=1 TO 4
120       PRINT T;
130     NEXT T
140 NEXT I
```

and so on. Neat, and fast.".

> END

```
_____
   VIEWDATA + TI-99/4A + C99 = PRESTEL
_____
```

By DAVE HEWITT. Contact Dave if you need to on (0865) 341428

This short article is a report of some of my recent work in putting together a Prestel system on a shoe string budget. The purpose of this is to share my thoughts in order to provoke thought from other TI users while at the same time being a short introduction to using Clint Pulley's C99 small c compiler.

This saga started when a good friend of mine (an ex ITUGer) purchased a Decafax Viewdata terminal. At first I did not know what this was or what it did but I soon realised its potential.

The unit in question is a Decafax VP1 viewdata plinth unit. It contains almost everything required for accessing Prestel and many other similar systems via your telephone line. It contains an autodial modem operating at 1200/75 baud, full colour viewdata decoder with output for an RGB monitor plus all power supplies etc. You simply plug it into your phone socket, monitor and mains supply and you have an instant Prestel system. Considering that these units are available for £19.95 + postage I am sure you will realise that it represents extremely good value for money. I am not trying to advertise but merely think a good thing should be shared with others. I can supply details of the supplier to anybody that is interested. Note that I am NOT the supplier so please don't try to order one from me.

These units are not new, they were first made in 1980-ish. The first consequence of this is the first thing you will probably have to do is change the old type BT jack plug for the new type of telephone connector. Also there is a potential unreliability in one component as follows: The internal modem can store 4 telephone numbers in a non-volatile RAM IC.

These however are in some cases proving unreliable, the consequence of which is to corrupt the numbers when the unit is switched off. This is more an inconvenience since you merely have to re-program the memory and all is well again. To cure this problem it requires replacement but as yet I have failed to even find the device listed in any data books let alone find a supplier. The device is a TMS3529NL which is in a 16 pin plastic package and I believe it to be 4 bit by 64 word non-volatile RAM. If anybody knows anything about this device please let me know. I think the age and potential problems are adequately reflected in the price of the units.

Where do the TI-99 and small c enter the equation ??. Well, as standard, the VP1 only has a numeric keypad. Fine for looking at pages but no use for sending messages, and you will soon find that this represents a severe limitation to the system. The supplier of the VP1 will sell you an alphanumeric keyboard for £63. For this you get a small keyboard which although it has full travel keys they are far too small and closely spaced with very poor feel and hence typing is a chore. In short you pay a lot for a poor excuse for a keyboard. This is where the TI enters the picture.

>>MORE>

The keyboard is required to send data in serial bit form at TTL levels. The protocol is normally high (mark), one start bit, 7 data bits, even parity bit and one stop bit at 75 baud. This can simply be produced from an RS232 port with a small circuit to convert the voltage levels. (I will detail this in a future article if anybody is interested).

However my system is based around my DIY expansion box. I don't have an RS232 card, instead I use my home built Centronics printer card (still available if anybody is interested). This left me with a problem. How to send serial data without an RS232 card.

The answer was simple. Following on the work by BILL REED published in TI-LINES V4.6, I have extended this idea and written a C99 program that emulates a Viewdata keyboard using the joystick port as the output. This approach is in some ways more convenient than RS232 as no interface electronics are required.

I will deal first with the software. The program shown here is really quite simple and so is a good introduction to C99. I will not try to explain in detail the program operation but will summarise. The main function, after initialising some variables and printing a title enters an endless loop scanning the keyboard and sending the ASCII value of any key pressed. There are two special cases which are tested with IF statements. IF (c == 2) checks for FCTN 4 and if so exits from the loop and hence terminates the program. IF (c == 10) checks for the Enter key. If this is pressed then the value 95 is sent instead of 10. This is because on Prestel the hash key (which on Prestel is ASCII code 95) is used as a sort of Enter key to terminate input and it seemed logical to define the Enter key on the TI to send this code.

The function 'send' is the part of the program that actually transmits one character via the joystick port. Initially the parity count is set to 0. The FOR loop runs from 1 to 7. Each time through the loop the value to be sent is shifted right one bit and its LSB tested for high or low. If high the parity count is incremented and a high bit sent else a low bit is sent. An interesting point here is that we really want the loop to run from 0 to 6. This is not possible though since for some reason it is not possible to shift right 0 bits. This is why 'key' is shifted 1 bit left before entering the loop. At the end the LSB of the parity count is checked. This is a simple way to tell if it contains an odd or even value. If it is even then a low parity bit is sent else a high is sent. Lastly a high stop bit is sent.

The last two functions 'high' and 'low' are the Assembly language from Bill's article. The timings are modified for 75 baud by changing the value loaded into R6. These timings should be 8 times greater than in Bill's program but for some reason this produced pulses too short so the exact value was determined by measurement and trial and error.

To enter the program firstly you must type the source code using a text editor (I use Funnelweb V 3.4) and save it to disk. I used the filename 'DSK2.VIEWDATA;C'. Next run the c compiler. Press Enter for the default answer to the two opening questions then enter 'DSK2.VIEWDATA;C' for the input filename and 'DSK2.VIEWDATA;S' for the output filename. The compiler will now hopefully compile your program. Any errors found will be typing errors so carefully check the program.

2022 history note- Viewdata introduced services now common on the Internet but was very costly and TOTAL subscriber base never reached 100k. The steadfast subscribers ended up on AOL..ss

>>MORE)

Next you must run the Assembler. Specify 'DSK2.VIEWDATA;S' for the source filename and 'DSK2.VIEWDATA;O' as the object filename. For options enter 'RC'. The C manual says use no options but the C option generates compressed object code which uses less disk space and with this program several errors will be reported if the R option is not used.

Now if all is well we can test the program. Select the Loaders option from Funelweb then option 4 for E/A LOAD RUN type files. Enter 'DSK2.VIEWDATA;O' for object filename then 'DSK1.CSUP'. When asked for a third object filename press Enter. The program should now be loaded and the program link table should be displayed. Type START and press Enter and the program will (hopefully) run.

Assuming the program runs okay you will get an opening title. From then on anything you type will be both displayed on the screen and sent at 75 baud from the joystick port. Pressing FCTN 4 will terminate the program with the usual c99 exit/rerun question.

I admit that this program is not earth shatteringly complicated but does perform a useful function and is yet another practical use to which my TI is put. I am presently developing a bells and whistles version of this program that allows messages to be prepared off line, saved to disk and then sent by a single keypress. If anybody is interested I can supply this version when it is finished.

Once you are happy with the program you may soon get tired of loading it in the way described above. We can improve this by converting the program to a 'PROGRAM' format file. For this you need to use the SAVE utility on the Funelweb V3.4 disk. To do this select the E/A LOAD RUN (option 4) as before.

This time load the following files in this order: DSK1.C99PFI, DSK2.VIEWDATA;O, DSK1.CSUP, DSK1.C99PFF and DSK1.FWSAVE, then press Enter. Enter SAVE as the program link name. If all is well you will be asked for a filename to save the program under. I use the name 'DSK2.VIEWDATA'.

If all works you will now have the program saved in memory image format. From now on all you have to do to load the program is select option 3 E/A PROGRAM loader and enter the filename 'DSK2.VIEWDATA' and the program will load and then run in one quick operation. Much more convenient.

I hope this brief article will be of use to some of you, if not directly for viewdata, as a beginners guide to using c99. I have assumed throughout that you are using c99 running from Funelweb V3.4 under Extended Basic. The above procedure will need to be modified if you are using other forms of loader.

That deals with the software. Now the hardware bit. This is basically the same as in Bill's article but as the load we are driving is a TTL input and the polarity of the signal is correct there is no need for any buffers. Hence a purely passive connection is required. This is detailed separately.

I find it inconvenient to use the cassette port to obtain the earth connection so in my console I have connected pin 1 of the joystick
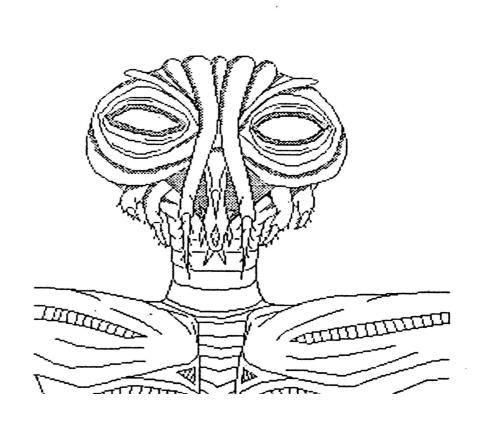
>>>MORE)

socket to 0v. This pin was previously not connected. If you are not happy doing this mod use the cassette port for the OV connection.

If anybody is interested in this program but has difficulty or can't be bothered to type it in then I may be able to supply a copy for reasonable copying costs. Contact me for details.

Some of you may be thinking that all of this performance to get a program running is not worth the effort. In this case it is true that a version of this program could be written for Extended Basic but c99 really comes into its own when program speed is critical and personally I prefer the C language to Basic.

Happy compiling, don't wear out your drives.   Dave Hewitt

>>>END

C99 source instructions supplied with David's article:

2022 Note: When you see a £ sign below,
instead use a # symbol.

```
/********************************************************************
 *                                                                  *
 *   VIEWDATA alphanumeric keyboard emulator. (C) D.G.Hewitt 30-12-1987 *
 *                                                                  *
 ********************************************************************/

/*------ main function. keyboard scan and transmit character. ------*/
main()
{
int c;

puts("VIEWDATA ALPHANUMERIC KEYBOARD EMULATOR\n\n\n");
puts("PRESS FCTN-4 TO ABORT       (C) D.G.Hewitt\n\n\n");

for(;;){
    c=getchar();                    /* read keyboard  */
    if (c==2) break;                /* exit on FCTN 4 */
    if (c==10) c=95;                /* if ENTER send '£' */
    send(c);                        /* send character */
    }
}

/*----------- function to send one character at 75 baud. -----------*/
send(key)
int key;
{
int loop,parity;
key = key << 1;
parity=0;

low();                                          /* send start bit */

for (loop=1;loop<8;loop++)              /* send 7 bits LSB first */
    {
    if (((key >> loop) & 1) == 1)           /* check each bit */
```

>>>MORE)

```c
        {
        parity++;           /* if high increment parity count */
        high();                             /* and send HIGH */
        }
    else
        low();                              /* else send LOW */
    }
                                /* now check parity */
if ((parity & 1) == 0)
    low();                      /* if even send LOW parity bit */
else
    high();                     /* else send HIGH parity bit */

high();                     /* stop bit to complete transfer */
}

/* ------- function to set O/P HIGH then wait for 13.333 mS -------*/
high()
{
£asm
        CLR R4
        LI  R5,)0606
        LI  R6,)0550
        LDCR R4,3
T1      DEC R6
        JNE T1
£endasm
}

/* ------- function to set O/P LOW then wait for 13.333 mS -------*/
low()
{
£asm
        CLR R4
        LI  R5,)0606
        LI  R6,)0550
        LDCR R5,3
T2      DEC R6
        JNE T2
£endasm
}
```

2022 note: When you see a £ symbol on these pages, instead use a # symbol. - due to the way Peter used his printer.

>>END

The following output from the C99 compiler is included to provide some degree of confirmation that you entered the C99 program correctly.

```
*c99 v2.1 (c) 1986 Clint Pulley
  REF C$CIND,C$DIV,C$REM,C$ASR,C$ASL,C$EQ,C$NE,C$LT,C$LE
  REF C$GT,C$GE,C$ULT,C$ULE,C$UGT,C$UGE,C$LNEG,C$SWCH
  REF GETCHA,GETS,PUTCHA,PUTS,LOCATE,POLL,TSCRN,EXIT
  DEF MAIN
MAIN
  DECT 14
  LI  8,C$1+0
  BL  15
```

>>MORE>

```
       BL *12
       DATA PUTS
       INCT 14
       LI 8,C$1+43
       BL 15
       BL *12
       DATA PUTS
       INCT 14
C$4
       B @C$5
C$2
       B @C$4
C$5
       MOV 14,8
       BL 15
       BL *12
       DATA GETCHA
       MOV *14+,9
       MOV 8,*9
       MOV 14,8
       MOV *8,8
       BL 15
       LI 8,2
       BL @C$EQ
       ABS 8
       JNE $+6
       B @C$6
       B @C$3
C$6
       MOV 14,8
       MOV *8,8
       BL 15
       LI 8,10
       BL @C$EQ
       ABS 8
       JNE $+6
       B @C$7
       MOV 14,8
       BL 15
       LI 8,95
       MOV *14+,9
       MOV 8,*9
C$7
       MOV 14,8
       MOV *8,8
       BL 15
       BL *12
       DATA SEND
       INCT 14
       B @C$2
C$3
       INCT 14
       B *13
C$1 BYTE 86,73,69,87,68,65,84,65,32,65,76,80
    BYTE 72,65,78,85,77,69,82,73,67,32,75,69
    BYTE 89,66,79,65,82,68,32,69,77,85,76,65
    BYTE 84,79,82,10,10,10,0,80,82,69,83,83
    BYTE 32,70,67,84,78,45,52,32,84,79,32,65
```

>>>MORE>

```
       BYTE 66,79,82,84,32,32,32,32,40,67,41,32
       BYTE 68,46,71,46,72,101,119,105,116,116,10,10
       BYTE 10,0
       EVEN
     SEND
      AI  14,-4
      MOV 14,8
      AI  8,6
      BL  15
      MOV 14,8
      AI  8,8
      MOV *8,8
      BL  15
      LI  8,1
      BL  @C$ASL
      MOV *14+,9
      MOV 8,*9
      MOV 14,8
      BL  15
      CLR 8
      MOV *14+,9
      MOV 8,*9
      BL  *12
      DATA LOW
      MOV 14,8
      INCT 8
      BL  15
      LI  8,1
      MOV *14+,9
      MOV 8,*9
     C$11
      MOV 14,8
      INCT 8
      MOV *8,8
      BL  15
      LI  8,8
      BL  @C$LT
      ABS 8
      JNE $+6
      B   @C$10
      B   @C$12
     C$9
      MOV 14,8
      INCT 8
      MOV *8,9
      INC *8
      MOV 9,8
      B   @C$11
     C$12
      MOV 14,8
      AI  8,6
      MOV *8,8
      BL  15
      MOV 14,8
      AI  8,4
      MOV *8,8
      BL  @C$ASR
      BL  15
```

>>MORE>

```
        LI 8,1
        INV *14
        SZC *14+,8
        BL 15
        LI 8,1
        BL @C$EQ
        ABS 8
        JNE $+6
        B @C$13
        MOV 14,8
        MOV *8,9
        INC *8
        MOV 9,8
        BL *12
        DATA HIGH
        B @C$14
C$13
        BL *12
        DATA LOW
C$14
        B @C$9
C$10
        MOV 14,8
        MOV *8,8
        BL 15
        LI 8,1
        INV *14
        SZC *14+,8
        BL 15
        CLR 8
        BL @C$EQ
        ABS 8
        JNE $+6
        B @C$15
        BL *12
        DATA LOW
        B @C$16
C$15
        BL *12
        DATA HIGH
C$16
        BL *12
        DATA HIGH
        AI 14,4
        B *13
HIGH
                CLR R4
                LI R5,)0606
                LI R6,)0550
                LDCR R4,3
T1              DEC R6
                JNE T1
        B *13
LOW
                CLR R4
                LI R5,)0606
                LI R6,)0550
                LDCR R5,3
```

>>> MORE)

```
T2       DEC R6
         JNE T2
  B *13
  END
```

>>>END

TI 99/4A

JOYSTICK PORT
PIN 7

CASSETTE PORT
PIN 3 (OR
JOYSTICK PORT
PIN 1 WITH OV
MODIFICATION)

OV

DATA

5 PIN 270°
DIN PLUG

1

5

VIEW DATA
PLINTH
KEYBOARD
SOCKET

VIEW DATA
PLINTH

TI99/4A - VIEW DATA VPI PLINTH
INTERCONNECTION LGAD.

```
    4   3
  5 •  • 2
  •  • 1
```

5 PIN DIN
PLUG PIN
NUMBERING.

LITTLE-ENDIANS AND BIG-ENDIANS
----

by GEOFF TROTT, originally published in TIsHUG NEWS DIGEST, produced by
the ILLAWARRA REGIONAL GROUP - TIsHUG

In the world of computers there are two classes of people: those who are
Little-endians and those who are Big-endians. These classifications
have nothing to do with anatomy, but rather with the prejudices people
have about the order in which the bits in data and addresses are
numbered. Just a storm in a tea cup you say? Well I guess that is
right, but the different ways used to number bits and bytes do cause
confusion, particularly to assembly language programmers, so I will
attempt to explain clearly what it is all about.

If you look at the circuit diagrams for the 9900 processor, or any
description of the instruction bit codes for example, you will notice
that the most significant bit of the 16 data bits or address bits is
numbered 0 and the least significant bit is numbered 15. This is the
pattern for the Big-endian order of the world.

Also, the first byte in the word, which is the even address, is also in
the most significant position. This means that if words are written
down in their numerical order across a page, the order of the bits and
bytes all start from the top left and read naturally to the right. The
pairs of bytes go together naturally to make words and the character
strings are in their correct sequence. This is all so logical that it
is hard to imagine why another way would be thought about or even
preferred.

Let us have a look at the Little-endian way of numbering bits and bytes.
The obvious difference is that the least significant bit is numbered 0
and the most significant bit has the largest number. The first byte
would also go into the least significant byte position. This means that
if we lay out the data in bytes starting from the top left of the page,
then the words are in the wrong order bytewise as it were, or if the
words are written down in their correct form, the bytes are not
consecutive.

Let us take an example of a character string of 5 characters, and assume
the format used by TI of the first byte containing the length of the
string.   The bytes would be 05 48 65 6C 70 21 in hexadecimal for the
string 'Help!'.   For the Big-endians this string would pack into
consecutive words like this. 0548 656C 7021.

For the Little-endians it would pack into words in the following way:
4805 6C65 2170. The same sort of problem arises with the Little-endian
way when 16 bit data is broken into 2 bytes. It is in low byte followed
by high byte order.

Well, why would anyone want to be a Little-endian? For those of us who
delve into the hardware, it is clear that the hardware manufacturers are
confirmed Little-endians. This is because the number of bits used by a
piece of hardware is generally less than that used by a computer.

>>>MORE>

Several pieces of hardware are connected in parallel to do the job. In this case it makes more sense to consistently number the least significant bit as 0, because this will be correct for at least one of the pieces (for a Little-endian). If you consider the address lines of memory chips, these can have anywhere from 8 to 16 address lines, and there may be families of these components which change only by an increase in address lines.

The Little-endian approach allows the address lines to be increased without requiring all the other address lines to be renumbered. This is the main advantage of the Little-endian approach, the ability to increase the number of address lines with the smallest impact on the numbering of the existing address lines. There are also advantages when transferring bytes to words and vice versa, as the bytes are in the correct part of the word to carry on normal arithmetic on the byte value. With the Big-endian approach the transfer of a byte into a word leaves the byte at the high end of the word, which then requires a byte swap within the word before normal arithmetic can be used - INC or DEC for example.

Some problems arise when the two concepts are mixed in the same system. This occurs within the design of some systems (DEC computers and the Motorola 68000 are examples), but it can occur in any system if the software is written accordingly. Most of the 99/4 computer was designed by Big-endians, but some of the disk operating system, was written by a Little-endian. It may be that the disk operating system is reasonably close to some other disk operating system from another computer, or just that whoever wrote it was a Little-endian at heart. This is why the file segment entry table appears mixed up, and the bit map is also in an odd order. So if you are delving into the disk operating system, or you find what appears to be an odd ordering of bytes in a word, brush up on your Little-endian thinking!

>>>END

```
-------------------------------------------------
T H E   T E A C H I N G   C O M P U T E R
-------------------------------------------------
```

By JIM PETERSON

I still consider the TI-99/4A to be a HOME computer, although many Users have expanded it far beyond that.

And what are the uses of a home computer? Primarily, entertainment and education - and, if you can justify the expense of an RS232, printer and modem, word processing and telecommunications.

The importance of the computer in education has been overemphasized in TV advertising, to the extent that people were offended by suggestions that, if they failed to buy a home computer, they were condemning their children to a life of failure. One man even formed an organization to oppose the TI-99/4A ads! However, the educational potential of the TI-99/4A was never realized. To teach any subject, a planned series of lessons is required. And, since each lesson is soon learned and no longer needed, the individual lessons must be inexpensive.

Texas Instruments did put out several educational modules, but they were not in any planned series and, in those days, they were not cheap. The Plato series is carefully planned, and excellently designed from an educational standpoint - but it does not take full advantage of computer capabilities, and is far too expensive.

Third party manufacturers did put out some fine educational modules but again, no planned series, and I do not know of any of them still supporting the TI.

In the early days, Micro-Ed and a few others put out some good educational software on cassette. As far as I know, Kidware and Tigercub are the only ones still offering educational programs on cassette.

There is also a vast amount of Public Domain software written by amateur programmers. It ranges in quality from mediocre to excellent, consists mostly of pre-school teaching or basic math drills, and would take much effort to organize into any course of education.

The market for educational software has been so poor that it is doubtful that any more will be produced. So, if you need it, you'll have to write it for yourself! Ideally, educational programs should be written by teachers, because they know how to teach. Unfortunately, it seems that few teachers have learned how to write programs, although some of their students have. A co-operative effort could have resulted in a large pool of good educational Public Domain software available to schools. However, the educational establishment as a whole has been so brainwashed by the Apple peddlers and the disciples of Logo that they will give no support to anything else.

What makes a good educational program?

If possible, it should be interactive - it should:

>>>MORE>

1. Teach a lesson;
2. Test to see if you have learned the lesson;
3. If you have not, go back and teach it again;
4. If you have learned, go on to the next lesson.

This is not always practical in a single program, but can be done by having one program run the next program.

I used a variation of this technique in "Casting Out Nines", which teaches a method of checking long multiplication and division problems. The student must correctly solve several problems at each step before the program will continue to the next step.

The program should give some kind of spoken, printed, graphics or musical reward for correct answers, and a corresponding admonition for wrong answers. The nature of these depends on the age level of the student.

In "Kindermath" I use a groan and a frowning face which can be changed to a smile and music by a correct answer. In "Kinderminus", the problem is displayed in the centre of a multicoloured kaleidoscope which changes patterns for each correct answer or turns black for a wrong answer. For an older student, a short game to be played could be offered after successfully answering a series of questions.

Computer graphics and sound should be fully utilized, but not allowed to become boring. The same musical salute after each correct answer soon gets tiresome. In some of my programs, a note is added to a tune for each correct answer, and after several such answers the entire tune is played. "Kindermath" uses several nursery tunes in succession.

The "stupid computer syndrome" should be avoided. This occurs when random selection causes the same question to be asked twice in succession. It is easily prevented by the simple statement IF Q=Q2 THEN (go back for another selection) ELSE Q2=Q.

The computer is especially well adapted to teaching Maths, because it can generate an infinite number of random problems to be solved. When the problem requires keying in an answer to a multi-digit problem displayed on the screen, input should be accepted from right to left in the same way as it was being worked on paper. My "Maths Homework Helper" does this with addition, subtraction and multiplication problems of any size, and also helps the student by refusing to accept an incorrect digit.

When possible, if a wrong answer is given the student should be shown how to work the problem. In one of our Extended Basic classes we analysed a Maths quiz program which generated random problems in the form of "IF 3 BOYS CAN CATCH 12 FROGS IN 4 DAYS, HOW MANY FROGS CAN 9 BOYS CATCH IN 8 DAYS?". If the answer was wrong, a screen display explained, "NO, THAT'S WRONG. IF 3 BOYS CAN CATCH 12 FROGS IN 4 DAYS, THEN 3 BOYS CAN CATCH 3 FROGS IN ONE DAY".....etc., through the problem.

Similar routines could be written for a wide variety of time/speed/distance problems, etc.

The basics of music education can also be easily taught by computer.

>>MORE>

The TI-99/4A can generate any musical tone required, and the piano keyboard, guitar fingering, musical notation, or whatever can be graphically displayed. John and Norma Clulow, Regena, and Bob Pomicter have written some excellent programs of this type.

Educational programs requiring much text are more difficult because of the limited memory capacity of the computer. Many of the Public Domain programs of this kind seem to have been typed in directly from a textbook, and there is really little reason for computerising them.

Some good module software is also of this type - nicely programmed, but very soon learned and discarded.

Most speed reading programs on the market are loaded with a data base of sentences, which are flashed on the screen briefly and the student is then asked to repeat them. He soon begins to recognize them from memory, even though he may think he is reading faster.

Also, the purpose of speed reading is to grasp the meaning of a sentence not its exact wording. My "Speeder Reader" and "Junior Speeder Reader" programs bypass these faults by being loaded with a wide selection of individual nouns, verbs, adjectives and modifiers which are randomly selected and combined into an infinite number of sentences. Then, the student is asked any one of several different questions about the sentence, requiring a one-word reply.

Public Domain spelling programs are popular but not very practical. In order to tell the student to spell a word, it must be either printed on the screen or spoken by the speech synthesizer. In the former case, he is already being shown how to spell it.

The vocabulary of the speech synthesizer is rather small. The TE II permits an unlimited vocabulary, but the pronunciation is not very clear - a sentence may be understood, but a single word out of context is more difficult. Ron Binkowski published a "Speller" program in the 99'er which overcame this problem by allowing a separate phonetic spelling of the word and an "as in" phrase which could be phonetically spelled until it sounded right.

In my "Miss Spell", I programmed each word in a correct and incorrect spelling, randomly showed either one to the student and asked him if it was correct and, if not, to spell it correctly. In "I & E Spelling" I programmed every word containing those difficult "ie" and "ei" combinations, randomly showed them on the screen with those two letters replaced by blanks, to be filled in. If the answer is wrong, the student is shown a screen displaying the "I BEFORE E EXCEPT AFTER C" rule with all its exceptions.

Finally, the best educational programs of them all are those that teach a person something while he thinks he is just having fun. Word games are of this type, whether the popular "Hangman" or "Scrabble" or whatever. My "Tirkle" is just a very simple little game for children, based on the early computer game called "Hurkle".

However, teachers have told me that they find it very useful for teaching young children logical thinking and compass directions.

The possibilities are endless - and so little is being done!

>>>END

A BASIC BOOK COLLECTION II

By PETER KILLICK

Pardon the stunningly original title, but it does make for easier indexing! The real reason for it is that since my first overview of some books for BASIC programmers, I have been loaned three more titles which deserve addition to that list, courtesy of a very kind ITUGer (who had better remain anonymous in case he gets inundated with requests!).

BEST OF 99ER VOLUME 1 (Out of print)

Contains most of the editorial from the first six issues of the original all-American TI magazine - five of them were cunningly deleted from the back number list by the publisher at around the exact time that he brought out this book.

Despite the rather restricted source for compilation, the book is well balanced and contains long and useful sections introducing Assembly language and LOGO to help you decide whether these are for you.

Shorter articles review software items like Extended BASIC and Programming Aids III and there are many programs for study or typing in, covering games (nothing really outstanding except for a particularly succinct version of every young monk's favourite pastime, "Tower of Hanoi"); and very good utilities that you can easily adapt to your own needs - "Micro Bartender"; a speech synthesizer vocabulary extender; computer mapping of spatial distributions; and "Electronic Home Secretary" being the best examples.

What I found REALLY encouraging was comparing this original version of Electronic Home Secretary with the republished version in Volume 4, issue 2 of "Home Computer Magazine" (99er's successor), which shows that even the genii who create all these published programs are still very much climbing a learning curve. The later version gives more comprehensive on-screen guidance yet is 15% shorter and the sort routine has been condensed from 64 lines bulging with 6 GOSUBs and 4 GOTOs to 24 lines with only one GOSUB still left!

COMPUTE!'s TI COLLECTION VOLUME 1 (COMPUTE! PUBLICATIONS INC.)

Another publisher whose ambitions for a mammoth series were dashed by the demise of the 99s - strange in this case as this book was published at least a year after TI's withdrawal from the market. Articles date from Cheryl Regena's first published feature for Compute! magazine in December 1982, and some parts of the book contain material by others that usefully reinforces her "Programmer's Reference Guide" mentioned last time (TI-LINES V4.4): e.g., a study of the 99/4A's character set, more ways to use (and understand!) CALL KEY(), and a primer on the use of sprites.

>>>MORE>

Unfortunately, not all of the many authors who contributed to this book are not up to Regena's standard. I was particularly caught out by the individual who GOSUBbed and GOTOed a lot of blank REM lines in his TI-WORD word processing program, so that when I deleted all the empty REMs and RESEQUENCEd the program I was left with an enormous task of tracing through all the proper new line references and replacing a myriad GOTO/GOSUB 32767 entries.

Especially putrid programming on his part in an item intended for publication, as I had carefully left in all the REMs that contained subprogram header references.

The book does contain a well-selected group of programs for home office management and the useful aspect is that these are all suitable for an unexpanded, cassette-driven, system, even the afore-dreaded word processor. There is a mini database manager written in two parts to suit the basic 16K memory, but numbered so that it can readily be MERGEd into one by those with Extended BASIC and 32K; a Mailing List that allows up to 45 addresses per cassette file; and Easycalc, a spreadsheet that provides an adequate 10 columns by 15 rows with the basic machine, or a generous 14 columns by 26 rows if you have the Extended BASIC/32K combination.

Oddly, the book version of Superfont, Patrick Parrish's excellent character-defining utility, is for an expanded system only (if anyone needs it, I have the magazine version which is for the 16K machine).

The difference between the two is worth having, though, as it is a "DOODLE" command which permits you to play around with your redefined set and create and save drawings made with it - the basic machine will not allow for this.

Some of the other subjects which are well covered in this Compute! book are: using a printer with the RS232 interface; making visual display presentations; animation with TI BASIC; sound creating and shaping programs; disk management utilities; and nine games, two of which require the Extended BASIC module.

All told, a book that would earn its keep in any collection unless, perhaps, you already have the same publisher's "Extended BASIC Home Applications" AND Regena's "Programmer's Reference Guide".

32 BASIC PROGRAMS FOR THE TI-99/4A (Dilithium Press)

A really good book for the TI newcmer, with the single proviso that many of the programs will be of most interest to those with mathematical abilities and inclination. I have neither, but still gained much insight from a book which is far more than just another collection of trite games - for example, two very relevant real-life computer applications are covered (statistics and simulations). The Statistics program does more than one which is in the more heavyweight-appearing Compute! Collection book, and although I couldn't follow the rationale of the PI program (which produces random "darts throws" to simulate data input for evaluating PI) that's entirely due to my learning disability in that area, and no fault of the book's authors.

>>>MORE)

The programs are nicely written and all are for the 16K machine, though you will need Extended BASIC for one of the games and one of the graphics demonstrations.

All the programs are fully explained with a description of their purpose, instructions, suggested changes and further projects (very good for learning purposes). To help you with the latter, there are complete summaries of all the routines and variables used.

Some care is needed when typing as the print quality of the listings is poor, with not much difference between the capital O and the zero, or the N and the M, and the computer's quirky idea of a multiplication sign here resembles a regurgitated currant more closely than ever.

>>>END

ROMOX SOFTWARE: The following items are to be made available through
ITUG to existing subscribers ONLY.

This is NOT a commercial operation, and vast quantities of profit are
not anticipated (nor are even small quantities!!!).

You may submit ROMOX previously purchased, or order up to FOUR ROMOX
already programmed with software of your choice from this list.

If you submit ROMOX previously purchased, there is no limit to the
number which you may have programmed.

There is a fixed fee of £5 per cartridge programmed, whether one or
several ROMOX are submitted for programming, or are ordered (subject to
current limitations).

Further lists may appear in due course, and the fee for programming may
vary from list to list. Due to the "soft" nature of this service, all
titles should always be available, unless withdrawn for whatever reason.

Further projects based upon the ROMOX cartridge are anticipated, and
details will appear as appropriate in due course.

```
----  --------------------------------------------------------------------
 DI    FIRMWARE
CODE   TITLE
----  --------------------------------------------------------------------
0001  ST NICK
0002  ASTROBLITZ
0003  MIDNITE MASON
0004  TOPPER
0005  HOPPER
0006  AMBULANCE
0007  FACE CHASE
0008  LOG JUMP
0009  BEYOND PARSEC
0010  ANT EATER
0011  CROSSFIRE
0012  DRIVING DEMON
0013  DEFENDER
oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
0014  SHAMUS           PLEASE DELETE FROM THE LIST - NO LONGER AVAILABLE
oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
0015  RABBIT TRAIL
0016  CAVE CREATURES
0017  CENTIPEDE
0018  EDITOR/ASSEMBLER
0019  STARTRAP
0020  STARGAZER 1
0021  STARGAZER 2
0022  STARGAZER 3
0023  D STATION 1
0024  D STATION 2
0025  TILE BREAKER
0026  KING OF THE CASTLE
0027  BOXER
----  --------------------------------------------------------------------
```

>>END

FOR SALE: packs of TEN disks, double-sided, double density, 96 tpi, but
can be used at lower capacities: £7 inclusive of post and packing.   If
you want disk boxes, I can now supply those WITH packs of ten disks at
£8.50 inclusive.   If you just want an empty disk box (technically called
a "library case"), I will supply them at £2.50 inclusive of post etc.

Contact ITUG (0865 510822 or write).

FOR SALE: MAXELL lockable disk boxes. 80/100 capacity: £16, 50 capacity
£10. Prices do not include post and packing. Contact ITUG (0865 510822
or write).

WANTED: Mr H CULPECK has asked if anyone can help him to obtain data on
the TRIUMPH-ADLER TA-1000 BUSINESS SYSTEM. If anyone can point him in
the direction of circuit diagrams, I/O details, and so on, he would be
very grateful. The configuration consists of cpu with 12K RAM, two 8"
floppy drives, an alphanumeric keyboard, and a printer. The problem,
apparently, is that no source code appears able to be loaded without a
punched card system (expensive) and the system will not run. Mr Culpeck
cannot even tell if he might load from tape deck. The last TA-1000 was
made in 1978, so it is not as if useful data is thick on the ground.

If anyone can help at all, please contact Mr Culpeck at 39, Carlton Way,
CAMBRIDGE, CB4 2BY.

WANTED: GERRY COLLINS urgently needs to purchase (or discuss the content
of) the owners manual and/or schematics for the TI PHP1700 STANDALONE
RS232 interface.

Contact Gerry on BOURNEMOUTH (0202) 433704 if you can help.

FOR SALE: Numerous MODULES. Those prefixed (U) are used but in good
condition; those prefixed (N) are new.

(N) MINUS MISSION	£ 5	(U) INDOOR SOCCER	£ 3
(U) VIDEO CHESS	£15	(U) HOUSEHOLD BUDGET MANAGER	£ 2
(U) HUNT THE WUMPUS	£ 5	(N) BEGINNING GRAMMAR	£ 4
(U) ADVENTURE (PIRATE)	£10		
(U) ADVENTURE (ADVENTURELAND)	£10	Further Items:	
(U) PERSONAL RECORD KEEPING	£10	(U) TWIN CASS. LEADS + ADAPTOR	£ 5
(U) MINIMEMORY	£20	(U) TWIN JOYSTICKS	£15
(N) EDITOR/ASSEMBLER WITH MANUAL	£25	(U) GAME WRITERS' PACK 2	£ 2
(U) EDITOR/ASSEMBLER WITH MANUAL	£15		
(N) TI-WRITER WITH MANUAL	£25	Various books are listed overleaf	
(U) EXTENDED BASIC	£20		
(N) TAX/INVESTMENT RECORD KEEPING	£ 9		
(U) MUSIC MAKER	£12		
(U) PERSONAL REPORT GENERATOR	£ 5		
(U) EARLY READING	£ 4		

)))MORE)

INMAN, ZAMORA, ALBRECHT		PECKHAM	
(U) INTRODUCTION TO TI BASIC	£ 5	(U) PROG. BASIC WITH THE TI	£ 5
TOWNSEND		TI PUBLICATIONS	
(U) LEARNING TO USE THE TI	£ 2	(U) BEGINNER'S BASIC	£ 4
WATT		AHL	
(U) LEARNING WITH LOGO (370pp)	£10	(U) 101 BASIC COMPUTER GAMES	£ 3
(For Apple II, with TI Appendices)		(Std. BASIC, usable with TI)	

Contact ITUG (0865 510822 or write). First come, first served.

All orders over £15 are post-free; otherwise, add £2.50 for postage.

I am now able to offer EPSON MODEL 540 AND 541 HALF-HEIGHT 40 AND 80 TRACK DRIVES at £95 per drive EXCLUDING post and packing. There are a limited number of these available, so I must sell them on a first come, first served basis.

Ring ITUG (0865 510822) or write to stake a claim to one (or two), with payment required fairly promptly, please.

>>>END